UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERÍA

MAESTRÍA EN GESTIÓN ESTRATÉGICA DE TECNOLOGÍAS DE LA INFORMACIÓN

"Integración de fuentes de datos heterogéneas: aplicando Tecnologías Semánticas y un Bus de Servicio Empresarial"

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGÍSTER EN GESTIÓN ESTRATÉGICA DE TECNOLOGÍAS DE LA INFORMACIÓN

AUTOR:

Ing. Johnny Cristian Solórzano Zambrano
CI: 0301432324

DIRECTOR:

Ing. Víctor Hugo Saquicela Galarza, PhD.
CI: 0103599577

CUENCA-ECUADOR

2017

Resumen

Actualmente las organizaciones disponen diferentes sistemas de información para soportar su modelo de negocio. Generalmente la incorporación de éstos sistemas se ha realizado sin seguir un proceso riguroso para determinar la interoperabilidad con los sistemas pre existentes, por otra parte la aparición de nuevas tecnologías han llevado a adoptar diferentes plataformas de software lo que ha ocasionado que se tengan islas de información dentro de una misma organización. En este entorno, las organizaciones se encuentran limitadas para la explotación de datos, la integración de procesos, así como en la producción de conocimiento e información para toma de decisiones. Los departamentos de Tecnologías de Información permanecen concentrados en tareas de mantenimiento y soporte, o en construcción de mecanismos de integración punto a punto, dejando a un lado su misión de apoyo a la estrategia de negocio.

Como caso de estudio se tomará a la Universidad de Cuenca, organización que actualmente cuenta con dieciocho sistemas informáticos, que en su mayoría operan de manera autónoma, esto implica que muchos procesos se encuentren desacoplados y la extracción de información se convierte en un verdadero reto puesto que el traspaso de datos entre aplicaciones se realiza utilizando scripts, archivos de texto y en algunos casos de forma manual lo cual implica pérdida de tiempo, riesgo de datos inconsistentes y por supuesto sobrecarga de horas de trabajo tanto para personal técnico como administrativo.

Basado en la problemática descrita, la principal contribución de este trabajo consiste en proponer un modelo de integración utilizando Tecnologías Semánticas y un Bus de Servicios Empresarial, que permita tener una visión unificada de las fuentes de datos en todas las operaciones transaccionales y de consulta.

Palabras Clave: ESB; Ontologías; Integración de Datos; Integración Semántica

Abstract

Organizations now have different information systems to support their business model. Generally the incorporation of these systems has been carried out without following a rigorous process to determine the interoperability with the pre existing systems, on the other hand the appearance of new technologies have taken to adopt different software platforms what has caused that they have islands of information within the same organization. In this environment, organizations are limited to data mining, process integration, as well as the production of knowledge and information for decision making. Information Technologyes departments remain focused on maintenance and support tasks, or on the construction of point-to-point integration mechanisms, leaving aside their mission of supporting business strategy.

It is common to find computer systems that operate autonomously, this implies that many processes are decoupled and the extraction of information becomes a real challenge since the transfer of data between applications is done using scripts, text files and in some cases manually which implies loss of time, risk of inconsistent data.

Based on the described problem, the main contribution of this work is to propose an integration model using Semantic Technologies and Enterprise Service Bus, which allows a unified view of data sources in all transactional and query operations.

Keywords: ESB; Ontologies; Data Integration; Semantic Integration.

Índice de Contenido

Índice de Figuras

Acrónimos Utilizados

API: Application Programming Interface
BPEL: Business Process Execution Language
BPM: Business Process Management
CLR: Common Language Runtime
DBMS: Data Base Management System
DFDL: Data Format Description Language
EIA: Enterprise Application Integration
EIPs: Enterprise Integration Patterns
ESB: Enterprise Service Bus
HTTP: Hypertext Transfer Protocol
IDE: Integrated Development Environment
JSON: JavaScript Object Notation
MOM: Message Oriented Middleware
MVC: Modelo Vista Controlador
NOR: Recursos No Ontológicos
OSGI: Open Services Gateway initiative
OWL: Web Ontology Language
RDF: Resource Description Framework
SOA: Service Oriented Architecture
SOAP: Simple Object Access Protocol
W3C: World Wide Web Consortium
WSDL: Web Services Description Language
XML: Extensible Markup Languajes

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

JOHNNY CRISTIAN SOLORZANO ZAMBRANO en calidad de autor/a y titular de los derechos morales y patrimoniales del trabajo de titulación "INTEGRACIÓN DE FUENTES DE DATOS HETEROGÉNEAS: APLICANDO TECNOLOGÍAS SEMÁNTICAS Y UN BUS DE SERVICIO EMPRESARIAL", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 13 de noviembre de 2017

JOHNNY CRISTIAN SOLORZANO ZAMBRANO

C.I: 0301432324

Cláusula de Propiedad Intelectual

JOHNNY CRISTIAN SOLORZANO ZAMBRANO autor del trabajo de titulación "INTEGRACIÓN DE FUENTES DE DATOS HETEROGÉNEAS: APLICANDO TECNOLOGÍAS SEMÁNTICAS Y UN BUS DE SERVICIO EMPRESARIAL", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 13 de noviembre de 2017

JOHNNY CRISTIAN SOLORZANO ZAMBRANO

C.I: 0301432324

DEDICATORIA

A mi querida esposa y mis queridos hijos por su aliento y positivismo que me impulsaron para la consecución de esta nueva meta en mi carrera profesional.

A mis padres por el apoyo brindado para cursar mis estudios de maestría.

Johnny Solórzano Z.

,

AGRADECIMIENTO

Expreso mi más sincero agradecimiento a todas las personas que compartieron su conocimiento y apoyo para la realización de este trabajo de titulación, de manera especial a mi director Ing. Víctor Saquicela, PhD por su acertada y oportuna asesoría.

Extiendo también mi agradecimiento al cuerpo docente y directivo del programa de Maestría en Gestión Estratégica de Tecnologías de la Información de la Universidad de Cuenca.

Johnny Solórzano Z.

1. INTRODUCCIÓN

De acuerdo a Beyer, et al. (2016), la integración de fuentes de datos cobra importancia precisamente por la existencia de un gran volumen y variedad de datos. Por lo tanto, poder gestionar adecuadamente esa información e interpretarla correctamente otorga una ventaja competitiva más que necesaria en la actualidad. La integración de fuentes de datos pretende definir arquitecturas, modelos e infraestructura de software que permitan a los usuarios acceder a datos almacenados en fuentes de datos heterogéneas, presentando una vista unificada de los sistemas de información. Uno de los grandes problemas a la hora de transformar la información disponible en conocimiento es la diversidad de estructuras y formatos de la información de partida. Para conseguir la mayor eficacia, es necesario integrarla bajo una estructura común homogénea como paso previo a su empleo.

Conseguir que aplicaciones distribuidas, heterogéneas y posiblemente autónomas colaboren entre sí, es un gran reto en el área de la integración de fuentes de datos. En este contexto, distribución, hace referencia a las diferentes aplicaciones que pueden ejecutarse en máquinas conectadas a través de una red (LAN o Internet). Autonomía, se refiere a que el sistema de integración no puede esperar que la aplicación cambie su forma de actuar para facilitar la integración (aplicaciones heredadas, aplicaciones de otros departamentos, aplicaciones de otras empresas, etc.). Finalmente, heterogeneidad, hace referencia a los diferentes tipos de software o hardware utilizados y sobre los cuales se ejecutan las aplicaciones.

El presente trabajo pretende: i) caracterizar los problemas de la integración de la información en general, ii) identificar las propuestas descritas en la literatura sobre el proceso de integración de fuentes de datos heterogéneas y iii) proponer una solución al problema de integración en las organizaciones utilizando tecnologías semánticas y un ESB[1].

[1] ESB- Bus de Servicios Empresarial, por sus siglas en inglés Enterprise Service Bus

El documento está organizado de la siguiente forma: en la Sección 2, se plantean los objetivos y alcance del trabajo. En la Sección 3, se describe brevemente el problema de la integración de datos, en donde se caracteriza las principales dificultades, enfocándose en los diferentes niveles de heterogeneidad que deben ser considerados al momento de proponer una solución, por otra parte, se realiza un análisis teórico de las alternativas para abordar los problemas de integración, además se describen algunos trabajos relacionados. En la Sección 4, se presenta un marco teórico tomado como base para el desarrollo de este trabajo. En la Sección 5, se describe el escenario analizado sobre el que se construirá la propuesta. En la Sección 6 se describen los componentes de integración utilizando tecnologías semánticas y ESB. En la Sección 7 se presenta la arquitectura de integración recomendada. Finalmente, en la Sección 8 se plasman algunas conclusiones de este trabajo.

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Definir una arquitectura prototipo para la integración de diversas fuentes de datos de la Universidad de Cuenca utilizando un Bus de Servicios Empresarial y Tecnologías Semánticas.

2.2 OBJETIVOS ESPECIFICOS

- Analizar las diferentes formas de integración de fuentes de datos.
- Definir modelos ontológicos para la representación del problema planteado.
- Construir y probar componentes dentro del ESB que permitan evidenciar la solución.
- Demostrar el proceso de integración con un requerimiento típico que encontramos al integrar fuentes de datos.

2.3 ALCANCE

Como resultado del análisis de las diferentes fuentes de datos que maneja la Universidad de Cuenca, se procederá a definir un prototipo de arquitectura de integración basada en ontologías y un bus de servicios empresarial. A través de un ejemplo se describirá cada una de las fases del proceso de integración. En este caso puntual se trabajará con un modelo entidad relación que típicamente se presenta en los sistemas de información demostrando la aplicabilidad de la arquitectura planteada.

3. ANTECEDENTES Y TRABAJOS RELACIONADOS

En esta sección se describen los conceptos más relevantes que se aplican en la integración de fuentes de datos como soporte de la propuesta, además se citan algunos trabajos relacionados al tema abordado en el presente documento, y se plantea la problemática que se pretende solucionar.

3.1. ANTECEDENTES

Lenzerine (2002) define la integración de datos como "El problema de combinar datos residentes en diferente fuentes y proveer al usuario una vista unificada de esos datos". Actualmente, tanto empresas medianas como grandes mantienen datos de manera heterogénea, procedentes de un amplio conjunto de fuentes, véase Figura 1. De acuerdo a diferentes análisis presentados por Bernstein (2008) acerca de los sistemas de información empresariales, se puede desprender que estos no han sido diseñados para integrar datos o aplicaciones a posteriori. Por lo tanto, en este entorno no es posible integrar los sistemas de información empresariales existentes, si se tiene como meta presentar al usuario final un sistema de información unificado. Para obtener con éxito una vista homogénea sobre datos de diferentes fuentes en una organización dependerá de: el contenido y funcionalidad de los actuales sistemas de información, la clase de información que es manejada por los distintos sistemas (datos alfanuméricos, datos estructurados, semiestructurados o no estructurados), la intención de uso de los sistemas de información (solo lectura o acceso a escritura), y la disponibilidad de recursos (tiempo, recursos humanos, dinero, etc.).

Estos factores han sido tomados en consideración en este trabajo para proveer una solución que intente balancear el costo-beneficio de una plataforma que permita la integración de fuentes de datos en las organizaciones.

La integración de múltiples fuentes de datos en general, tiene como objetivo combinar las fuentes seleccionadas de manera que formen un todo unificado y ofrezcan a los usuarios la ilusión de interactuar con un solo sistema de información. Los usuarios disponen de una vista lógica homogénea de los datos

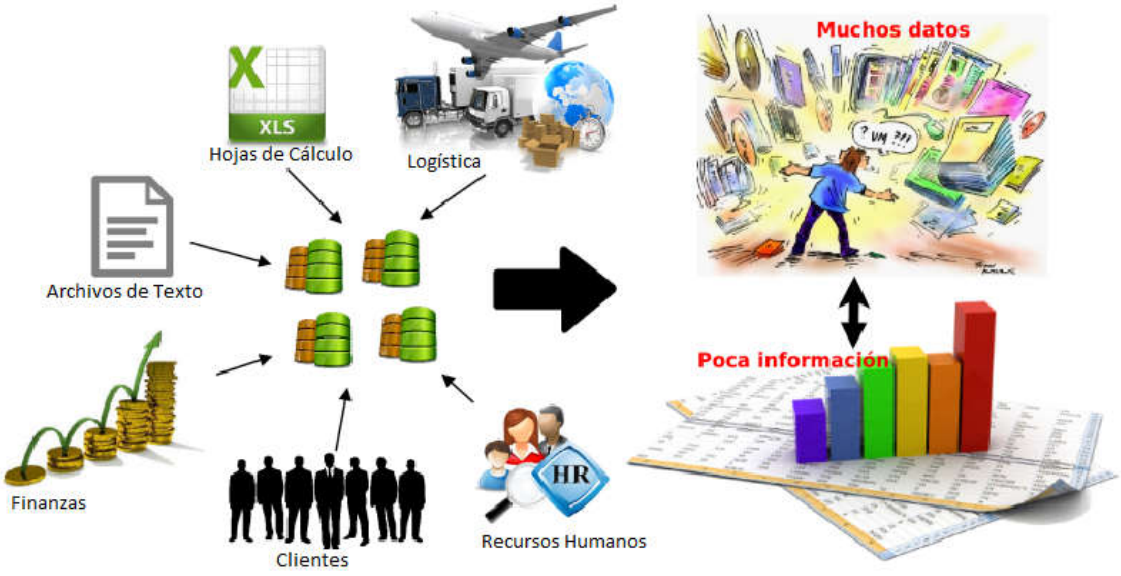


Figura 1. Problema de integración de fuentes de datos

que están físicamente distribuidos sobre fuentes heterogéneas. Para ello, todos los datos tienen que ser representados con los mismos principios de abstracción (modelo unificado de datos y semántica). Esta tarea incluye la detección y resolución de conflictos de heterogeneidad a múltiples niveles como:

- **Nivel de estructuración:** como está conformada la fuente de datos (información estructurada, información no estructurada y semiestructurada, información en formatos legibles para humanos pero no para máquinas).
- **Modelo de datos:** el tipo de modelo usado para describir los datos (modelo relacional, modelo jerárquico, otros).
- **Plataforma de software:** usada como herramienta, gestor de datos (Mysql, Postgresql, otros).
- **Convenciones de sintaxis:** diferentes maneras de expresar un mismo dato (Calle Bolívar, n◦ 2-37; C/Bolívar, 2-37).
- **Convenciones semánticas:** diferentes significados de entidades en la fuente de datos; tomando como ejemplo aplicaciones para gestión de nómina, una fuente de datos puede clasificar los rubros que intervienen

en un rol de pagos como “beneficios sociales”, otra fuente puede considerarlas como “sueldos y salarios” y “provisiones sociales”.

- **Diferencia de granularidad:** fuentes que tienen tablas con un campo para describir el nombre completo de una persona, otra fuente usa campos separados para almacenar el apellido y el nombre de la persona.

Para satisfacer las necesidades de integración que serán descritas en esta sección se analizan diferentes propuestas. La clasificación empleada para efectuar el análisis se fundamenta en el trabajo introducido por Dittrich (2000), en el que se distinguen propuestas de integración según el nivel de abstracción.

Los sistemas de información pueden describirse usando una arquitectura de capas, Tsierkezos (2010), como se muestra en la Figura 2. En la capa superior, los usuarios acceden a los datos y servicios a través de diversas interfaces que se ejecutan en la parte superior de diferentes aplicaciones. Las aplicaciones pueden utilizar un *middleware*, monitores de procesamiento de transacciones, *middleware* orientados a mensajes (MOM), SQL *middleware*, etc., para acceder a datos a través de una capa de acceso de datos. Los propios datos son gestionados por un sistema de almacenamiento de datos. Por lo general, se utilizan sistemas gestores de base de datos (DBMS) para combinar los datos de acceso y la capa de almacenamiento.

Integración Manual

En esta propuesta, los usuarios interactúan directamente con todas las fuentes de datos pertinentes y manualmente integran los datos seleccionados. Es decir, los usuarios tienen que tratar con diferentes interfaces de usuario y lenguajes de consulta. Además, los usuarios necesitan tener un conocimiento detallado sobre la localización, la representación lógica de los datos, y la semántica de los datos.

Interfaz de Usuario Común

En este caso, el usuario es provisto de una interfaz de usuario común (por ejemplo, un navegador Web) que proporcione una apariencia uniforme. Los datos de sistemas de información relevantes todavía se presentan por separado

de modo que la homogeneización y la integración de datos tiene que ser realizado por los usuarios (por ejemplo, como en los motores de búsqueda).

Integración de Aplicaciones

Este enfoque utiliza aplicaciones de integración que tienen acceso a diversas fuentes de datos y retornan resultados integrados al usuario. Esta solución es práctica para un pequeño número de sistemas a relacionar. Sin embargo, en la práctica, las aplicaciones son cada vez más numerosas como el número de interfaces de sistemas y formatos de datos para homogeneizar.

Integración por Middleware

El *middleware* proporciona funcionalidad reusable que se utiliza generalmente para resolver aspectos dedicados del problema de integración, por ejemplo, como lo ejecutado por el SQL *middleware*. El uso de esta propuesta implica el uso de diferentes herramientas de *middleware* las cuales por lo general necesitan ser combinadas para construir sistemas integrados.

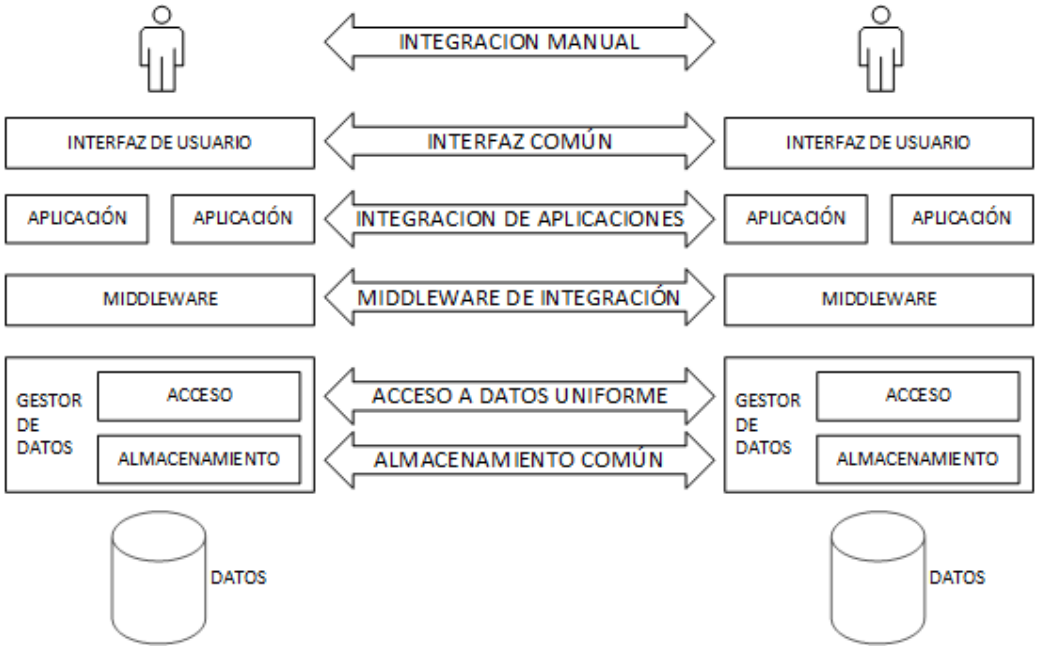


Figura 2. Tipos de Integración

Acceso Uniforme de Datos

En este caso, una integración lógica de los datos se realiza en el acceso a nivel de los datos. Aplicaciones globales son provistas de una visión unificada de la distribución física de los datos, aunque solo se dispone de datos virtuales en este nivel. Los sistemas de información locales mantienen su autonomía y pueden soportar capas adicionales de acceso a datos para otras aplicaciones. Sin embargo, una provisión global de acceso a datos integrados puede llevar mucho tiempo ya que el acceso, homogeneización, y la integración tiene que ser realizado en tiempo de ejecución.

Almacenamiento de Datos Común

Aplicando esta propuesta, la integración física de los datos se realiza mediante la transferencia de datos a un nuevo repositorio de almacenamiento, las fuentes de datos locales pueden ser retiradas o permanecer en funcionamiento. En general, la integración física de los datos proporciona un acceso rápido a los datos. Sin embargo, si las fuentes locales son retiradas, las aplicaciones que acceden a estas tienen que ser migradas al nuevo repositorio de almacenamiento también. Si se requiere que las fuentes de datos locales sigan funcionando un refrescamiento periódico del almacenamiento común de datos necesita ser considerado.

En la práctica, las soluciones concretas de integración se realizan sobre la base de los seis enfoques de integración presentados en los párrafos anteriores. Sin embargo, en este trabajo se usa un enfoque de acceso de datos uniforme utilizando tecnologías ESB y semánticas que provea un simple punto de acceso para acceder a las consultas de varias fuentes de datos. Un mediador que contenga un procesador global de consultas será empleado para enviar subconsultas a las fuentes de datos locales. Más detalles de esta propuesta se desarrollan en las siguientes secciones.

Se descarta el uso de las otras propuestas de integración debido a que en el caso concreto de las dos primeras opciones, estas representan un esfuerzo enorme de implementación, pero sobre todo no permite escalar la solución a

futuros sistemas que se integren en la organización. La integración usando middleware[2] no fue considerada como una opción puesto que esta solución requeriría la incorporación de diferentes herramientas para construir un sistema integrado. No obstante, dado la heterogeneidad de los sistemas actuales en las organizaciones representa un proyecto de muy alto costo y riesgo. Finalmente, la opción de integración mediante un repositorio común se presenta como una buena alternativa para los futuros sistemas de información que se integren a la organización, sin embargo la gran diversidad de datos y esquemas encontrados en los sistemas actuales hacen inviable esta solución por su costo y tiempo de implementación.

3.2. TRABAJOS RELACIONADOS

Roa-Valverde et al. (2008) proponen el uso de un ESB combinado con tecnologías de Web Semántica con la finalidad de crear un sistema "inteligente" que facilite la integración de datos. La idea es agregar un módulo semántico a la funcionalidad del ESB de manera que se puedan anotar semánticamente los objetos que son manejados dentro del Bus, esto proporciona a los artefactos implementados en el Bus la capacidad de "razonar", facilitando tareas como el descubrimiento y composición de servicios.

Shi et al. (2014) presentan una plataforma de integración basada en la teoría de ontologías y un Bus de Servicios. Su propuesta indica que la integración se puede lograr haciendo un mapeo entre los modelos ontológicos de datos para el efecto utilizan la técnica de mapeo de 3 capas; y que las nuevas demandas se pueden atender mediante la reorganización de servicios.

Jin et al. (2014) proponen el uso de tecnologías semánticas para mejorar las capacidades de un Bus de Servicios, ellos manifiestan que los Buses de Servicios actuales únicamente permiten describir los servicios que proporcionan de manera sintáctica lo cual imposibilita una mediación de servicios con semántica y el razonamiento sobre los datos a integrar. Por este motivo,

[2] software orientado a proporcionar conectividad, interoperabilidad o integración entre diferentes aplicaciones, normalmente distribuidas.

proponen la incorporación de anotaciones semánticas basadas en ontologías como un mecanismo para enriquecer y conciliar la semántica de los datos que circulan a través del Bus de Servicios.

Schratzenstaller et al. (2016) plantean establecer un método de apoyo para permitir que los datos generados por empresas PYMES[3] integren sus sistemas de forma flexible y automática utilizando servicios web a través de ESB. Con el objetivo de resolver las incompatibilidades entre los datos que deben intercambiarse entre las PYMES proponen la creación de una ontología como método de análisis de datos semánticos.

Harcuba et al. (2015) plantean una solución para integrar clientes livianos con el sistema de mensajería del ESB utilizando una API RESTful[4] basada en HTTP[5]. Para el efecto se dispone de una pasarela que convierte los mensajes ESB, cuyo contenido está codificado en RDF[6] según la ontología OWL[7], a requerimientos HTTP con contenido JSON[8] y viceversa. Este trabajo lo han realizado dentro del proyecto europeo ARUM[9]. El marco de trabajo se utiliza para simplificar la integración de interfaces de usuario para la supervisión y programación de la producción, la cual se realiza desde un teléfono inteligente o un navegador Web, teniendo la infraestructura del ESB como núcleo que aloja los principales componentes computacionales.

Los trabajos citados evidencian la tendencia actual para incorporar tecnologías semánticas en los procesos de integración de fuentes de datos heterogéneas, en el presente trabajo se pretende lograr una arquitectura consistente basada en

[3] Se conoce como PYMES al conjunto de pequeñas y medianas empresas que de acuerdo a su volumen de ventas, capital social, cantidad de trabajadores, y su nivel de producción o activos presentan características propias de este tipo de entidades económicas. Tomado de http://www.sri.gob.ec.
[4] Una API RESTful es un método que permite la comunicación entre un cliente basado en web y un servidor que emplea REST (Representational State Transfer)
[5] https://www.w3.org/Protocols/
[6] https://www.w3.org/RDF/
[7] https://www.w3.org/OWL/
[8] http://www.json.org/json-es.html
[9] http://www.almende.com/arum

ontologías y ESB, sustentada en estándares, cuya implementación se pueda alcanzar en un corto plazo.

3.3 PROBLEMÁTICA

Empresas grandes y medianas así como gobiernos han realizado grandes inversiones en sistemas informáticos, generalmente estas inversiones se han orientado a conseguir soluciones de funciones específicas tales como recursos humanos, manufactura, contabilidad, ventas, gestión académica, etc.; que individualmente contribuyen a la mejora de la productividad de la organización. Sin embargo, al momento de adquirirlas no se ha analizado la necesidad de integrarlas a los sistemas existentes (Bernstein, 2008). Así, aunque una organización pueda invertir ingentes cantidades de recursos en sistemas de información, cuando estos no consiguen integrarse, dejan una gran parte de los procesos de negocio sin ejecutarse de forma continua.

Permanecer con este modo de operación tiene un impacto directo en la competitividad de una organización. Desde una perspectiva empresarial, estas brechas por falta de integración generan enormes costos y pérdida de oportunidades; también interfieren con la consecución de las metas estratégicas que se fundamentan en la operación de la organización en su conjunto, y no sólo de ciertos departamentos o secciones[10].

Como caso de estudio se tomará la Universidad de Cuenca, organización que actualmente cuenta con dieciocho sistemas informáticos (véase Figura 3), algunos de éstos sistemas mantienen integración parcial entre sí y otros operan de forma aislada, esto implica que muchos procesos se encuentren desacoplados y la extracción de información se convierte en un verdadero reto puesto que la transferencia de datos entre aplicaciones se realiza utilizando scripts, archivos de texto y en algunos casos de forma manual lo cual implica

[10] Integración de aplicaciones empresariales. Una decisión de negocios. Microsoft 2002. Recuperado de https://www.microsoft.com/Argentina/downs/empresa_consolidacion/Integracion_aplicaciones_empresariales.doc

pérdida de tiempo, riesgo de datos inconsistentes y por supuesto sobrecarga de horas de trabajo para el personal técnico y administrativo.

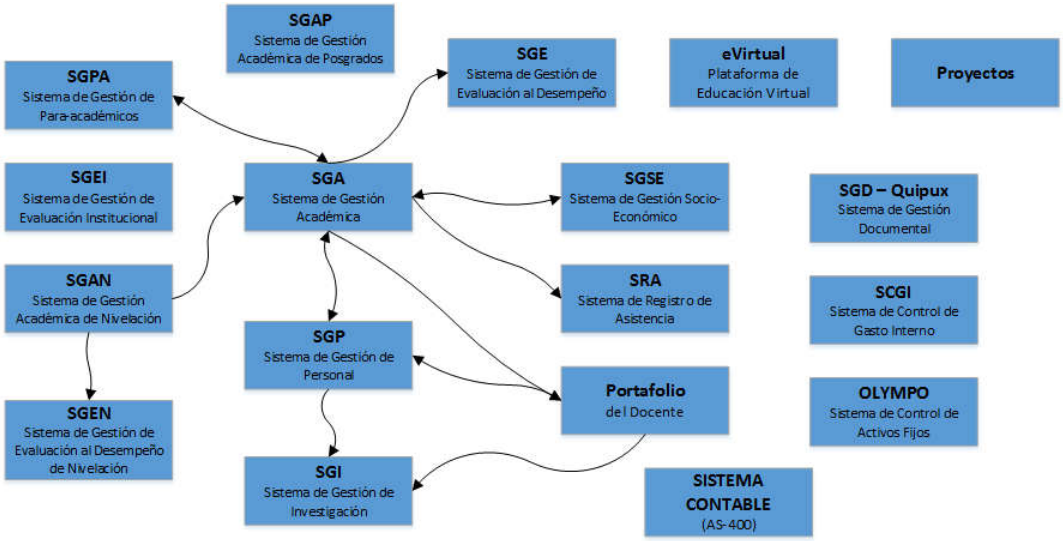


Figura 3. Sistemas Informáticos - Universidad de Cuenca

4. MARCO TEORICO

En este capítulo se describen los conceptos y tecnologías analizadas para ser aplicadas en la ejecución de este trabajo.

4.1 BUS DE SERVICIOS EMPRESARIAL - ESB

Un bus de servicios empresarial (ESB) es fundamentalmente una arquitectura, se compone de un conjunto de normas y principios para integrar múltiples fuentes de datos. Las plataformas ESB permiten a los usuarios construir arquitecturas de integración, pero varían en la forma en que lo hacen y las capacidades que ofrecen. El bus de servicios empresarial es una infraestructura de mensajería y comunicación construida sobre la arquitectura orientada a servicios (SOA)[11], sirve como un *middleware* que facilita la interoperabilidad de los servicios y aplicaciones heterogéneas dentro de las organizaciones. Técnicamente, el ESB por si solo permite la articulación de sistemas interactivos y permite distribuir la lógica de negocio de una solución en módulos incrementales, manteniendo su propio control local y la autonomía.

El concepto central de una arquitectura ESB es que permite integrar diferentes fuentes de datos, colocando un bus de comunicación entre ellas y permitiendo que cada fuente se comunique con dicho bus. Este hecho posibilita que los sistemas trabajen de forma desacoplada, con lo que se suprime la dependencia o el conocimiento de los otros sistemas en el bus. En las soluciones de integración punto a punto se crean dependencias estrechas entre aplicaciones, debido a que no existe centralización para controlar o solucionar problemas, lo que no le permite escalar, y se vuelve frágil y difícil de manejar en el transcurso del tiempo.

El concepto de ESB se originó por la necesidad de alejarse de la integración punto a punto y como una evolución de EAI[12] (Enterprise Application Integration). La incorporación de un ESB según IBM presenta las siguientes ventajas:

[11] http://www.opengroup.org/soa/source-book/soa/p1.htm
[12] http://searchmicroservices.techtarget.com/definition/EAI-enterprise-application-integration

- Mayor reutilización de activos de TI separando la lógica de las aplicaciones y las tareas de integración; con esto se reduce la cantidad, tamaño y complejidad de las interfaces de aplicación.
- Capacidad de añadir o cambiar servicios con una interrupción mínima en las aplicaciones.
- Reducir costos y riesgos involucrados en los cambios del negocio.

TIBCO Software Inc.[13] define a un ESB como un modelo de arquitectura de software que proporciona un acoplamiento flexible de servicios, y promueve la reutilización de fuentes de datos sin la necesidad de recodificarlas. Entre las capacidades comunes que brindan estas herramientas se incluyen transformación y mapeo de datos, colas y secuencia de mensajes y eventos, manejo de seguridad o excepciones, conversión de protocolos, etc.

Con una arquitectura de bus, todas las fuentes de datos siguen los mismos estándares y pueden compartir un método estándar de transferencia de datos entre ellas. Los ESB incorporan mecanismos para publicar y suscribir, lo que facilita que cualquier aplicación se conecte al bus, para el efecto deben ser compatibles con los estándares soportados por el bus. Un componente esencial de un ESB es la capa de mensajería. La gestión de mensajes mejora en gran medida la reutilización del servicio, porque pueden suscribirse a nuevos datos simplemente cambiando la configuración. La mensajería también ofrece una gran flexibilidad para las aplicaciones creadas con servicios, ya que otros servicios o clientes (como aplicaciones móviles) se pueden incorporar sin necesidad de modificarse. La infraestructura de mensajería que soporta el ESB aumenta la calidad de las aplicaciones puesto que se garantiza la entrega de mensajes entre los servicios. Por otra parte, se mejora la escalabilidad puesto que los datos se intercambian de manera más eficiente para millones de peticiones/respuestas.

[13] https://www.tibco.com/blog/2015/03/25/integration-broker-or-enterprise-service-bus/

El uso de un ESB, proporciona un enfoque altamente distribuido a la integración y brinda capacidades únicas permitiendo a los diferentes departamentos de una organización ejecutar sus propios proyectos de forma autónoma, sin dejar de conectar entre sí a cada componente cuando se requiere su de integración.

En la Figura 4, se muestra cómo se pueden conectar a un ESB: procesos de negocio, reglas de negocio, aplicaciones, manejo de documentos, datos, entre otros.

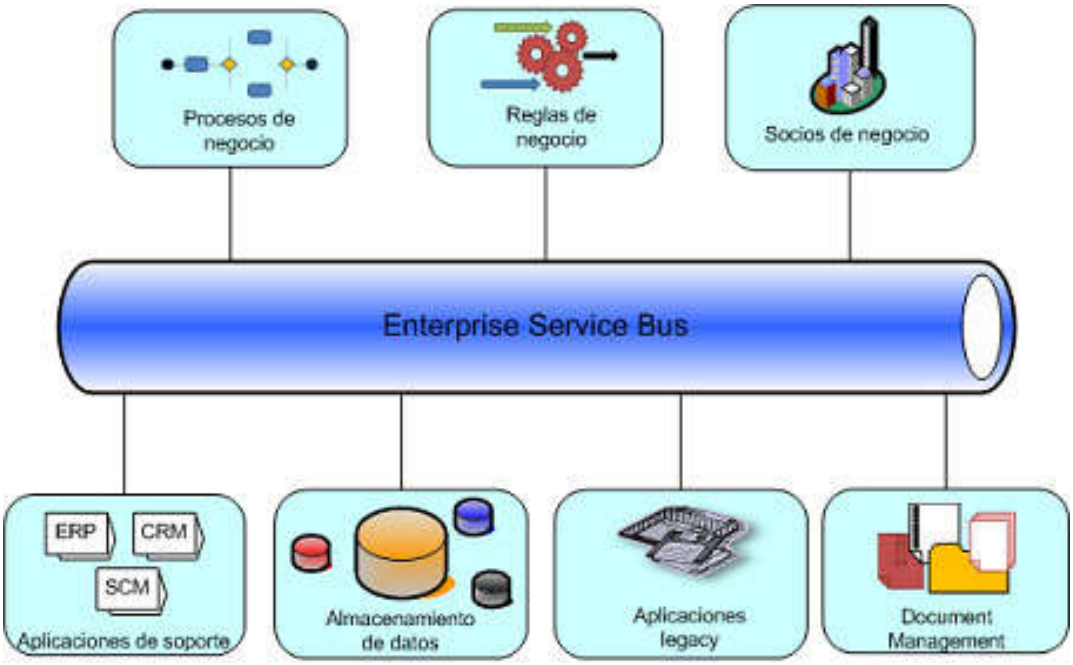


Figura 4. Diagrama de la Arquitectura ESB[14]

4.2.1 PLATAFORMAS ESB DISPONIBLES EN EL MERCADO

En la actualidad existe en el mercado una gran diversidad de plataformas ESB tanto propietarias como de código abierto, a continuación se presentan a manera de ejemplo las características de algunas de las herramientas que han sido seleccionadas por su popularidad y posicionamiento en el mercado.

[14] http://kaf.com.mx/home/font-styles-mainmenu-54

Oracle Service Bus (OSB)

Es un ESB (véase Figura 5) manejado por políticas y basado en configuraciones que proporciona una consola para la gestión de servicios dinámicos y políticas de configuración, así como también para monitoreo de sistemas y tareas. OSB facilita una arquitectura de acoplamiento flexible y centraliza el mantenimiento. La consola de esta herramienta está diseñada para dar respuesta rápida y de forma eficiente a los cambios en ambientes orientados a servicios.

A continuación se enumeran algunas de las características más destacadas de Oracle Service Bus[15]:

- Transforma y dirige dinámicamente los servicios utilizando reglas de enrutamiento simples y complejas.
- Orquesta servicios de los sistemas de TI existentes con diferentes protocolos de mensajería sin necesidad de cambiar el sistema y los estilos arquitectónicos.
- Aísla los cambios de ubicación de servicio.
- Responde rápidamente a las necesidades de la empresa configurando las reglas de enrutamiento basadas en cambios en las reglas de negocio existentes o sistemas de TI, sin necesidad de programación.
- Orquesta varios servicios para crear uno nuevo.

Oracle Service Bus aporta una infraestructura orientada a servicios tomando en cuenta las necesidades de la empresa convergiendo las capacidades de integración de un ESB con la gestión de servicios, para formar un solo producto; esto acelera la configuración y el despliegue, y simplifica la gestión de servicios compartidos sobre SOA.

[15] http://www.oracle.com/technetwork/middleware/service-bus/overview/index.html

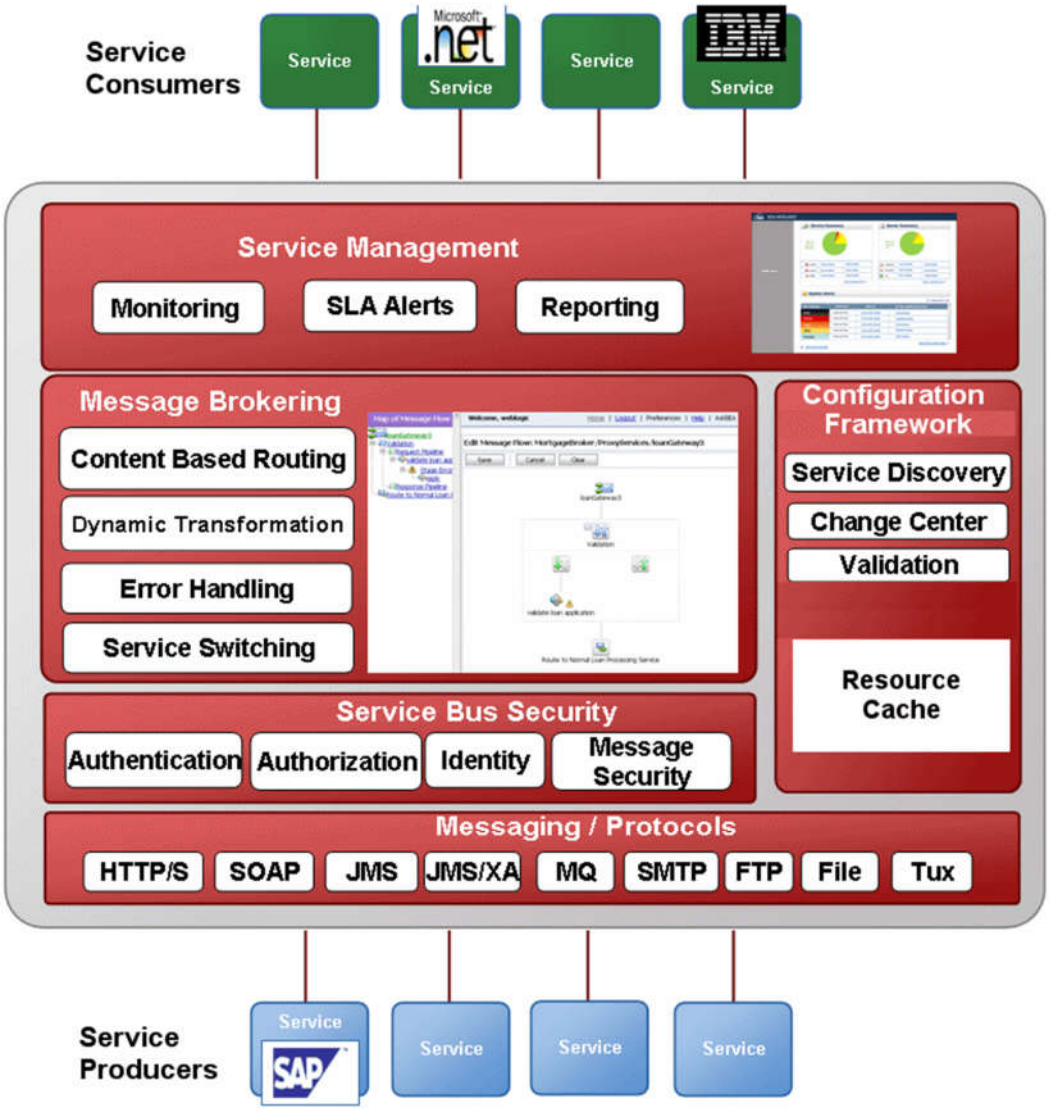


Figura 5. Arquitectura de OSB[16]

IBM Integration Bus

IBM Integration Bus Advanced[17] (véase Figura 6) es una plataforma de integración sólida y flexible basada en tecnología ESB. Proporciona conectividad y transformación de datos universal en entornos de tecnología heterogéneos. Entre las principales características de esta plataforma se pueden citar las siguientes:

[16] https://docs.oracle.com/html/E15020_01/architecture_overview.htm
[17] http://www-03.ibm.com/software/products/es/ibm-integration-bus

- Permite a empresas de cualquier tamaño eliminar las conexiones punto a punto y el proceso datos por lotes.
- Permite conectarse a una gran variedad de aplicaciones heterogéneas y servicios web, sin requerir una conectividad de punto a punto compleja.
- Proporciona un amplio soporte a aplicaciones y servicios basadas en plataformas Microsoft .NET[18] además provee un conjunto de herramientas de desarrollo integradas con Microsoft Visual Studio[19].
- Proporciona conectividad y transformación de datos universal para aplicaciones tanto basadas en estándares como las que no se basan en estándares.
- Permite definir reglas y aplicarlas a los datos o sucesos que circulan por WebSphere Message Broker[20].
- Posibilita el envío de información a otras aplicaciones para comparar datos casi en tiempo real con indicadores de rendimiento.
- Habilita el almacenamiento de datos altamente seguro para el análisis fuera de línea o el registro de auditoría sin necesidad de aplicar cambios en las aplicaciones de negocio.
- Permite conectarse a prácticamente cualquier aplicación o servicio sobre diversos protocolos, incluidos SOAP[21], HTTP o Java Message Service (JMS)[22], entre otros.
- Ofrece soporte directo a muchos de los idiomas soportados por Microsoft Common Language Runtime (CLR)[23].
- Incluye patrones para las transformaciones frecuentes que minimizan los errores y reducen el tiempo de implementación.
- Proporciona auditoría y diagnóstico mejorados mediante un navegador.

[18] https://www.microsoft.com/net/
[19] https://www.visualstudio.com/es/
[20] https://www.ibm.com/support/knowledgecenter/es/SSKM8N_8.0.0/com.ibm.etools.mft.doc/bb43020_.htm
[21] https://www.w3.org/TR/soap/
[22] http://docs.oracle.com/javaee/6/tutorial/doc/bncdq.html
[23] https://msdn.microsoft.com/es-es/library/8bs2ecf4(v=vs.110).aspx

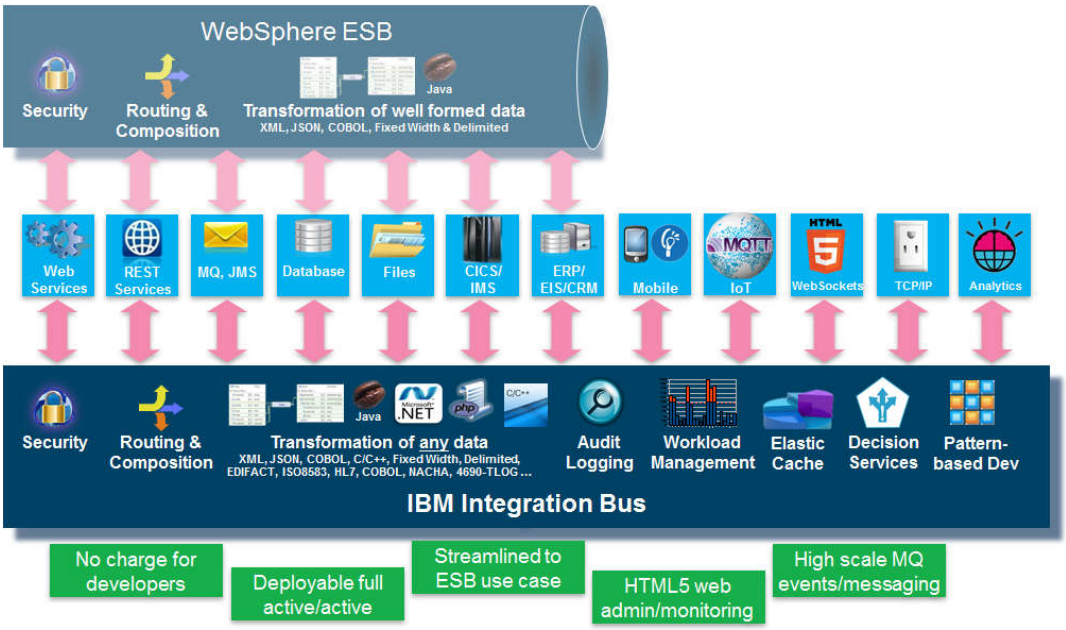


Figura 6. Arquitecura IBM Integration Bus

Apache Servicemix

Apache ServiceMix[24] es un contenedor de integración flexible, de código abierto que unifica las características y funcionalidades de Apache ActiveMQ[25], Apache Camel[26], Apache CXF[27] y Apache Karaf[28] en una plataforma para construir soluciones de integración. Proporciona un ESB de código abierto bastante completo, a continuación se muestran los diferentes componentes de software utilizados en la implementación del ESB:

- Fuse ESB - Apache ServiceMix. El ESB Apache ServiceMix incorpora el estándar OSGi[29] para la implantación de soluciones SOA *open source*. Apache ServiceMix se articula en torno a 3 proyectos diferenciados que funcionan coordinadamente para permitir las características propias de un ESB en un entorno OSGi:
    - Apache ServiceMix Kernel
    - Apache ServiceMix NMR
    - Apache ServiceMix Features

[24] http://servicemix.apache.org/
[25] http://activemq.apache.org/
[26] http://camel.apache.org/
[27] http://cxf.apache.org/
[28] http://karaf.apache.org/
[29] https://www.osgi.org/

El *kernel*, es una implementación básica de un contenedor OSGi basada en Apache Felix[30] al que se le han añadido ciertas funcionalidades y características para hacerlo más versátil. El NMR no es más que un conjunto de funcionalidades desplegables sobre cualquier contenedor OSGi, especialmente sobre el propio ServiceMix Kernel. En el NMR reside el bus de servicios ServiceMix así como la implementación del estándar JBI[31]. Es importante hacer notar que, si bien el estándar JBI se sigue soportando en ServiceMix, ya no se trata de la única alternativa posible a la hora de desarrollar soluciones SOA con ServiceMix.

A continuación se citan brevemente los diferentes componentes de software utilizados en la implementación del ESB (véase Figura 7):

- Fuse Service Framework → Apache CXF. Apache CXF es una plataforma completa, de código abierto para soportar servicios web.

- Fuse Mediation Router → Apache Camel. Apache Camel es un potente componente open source de integración de sistemas, diseñado para implementar los Enterprise Integration Patterns (EIPs).

- Fuse Message Broker → ActiveMQ. Apache ActiveMQ es un *broker* de mensajería de código abierto que implementa la especificación de Java Message Service 1.1 (JMS). Ofrece "características empresariales" tales como *clustering*, múltiples almacenes para mensajes, así como la capacidad de emplear cualquier administrador de base de datos como proveedor de persistencia.

- Fuse IDE →Eclipse. Es una herramienta gráfica basada en la herramienta de desarrollo Eclipse que permite integrar componentes que trabajan con Apache ServiceMix, Apache ActiveMQ, Apache Camel, y las distribuciones FuseSource.

[30] http://felix.apache.org/
[31] https://www.jcp.org/en/jsr/detail?id=208

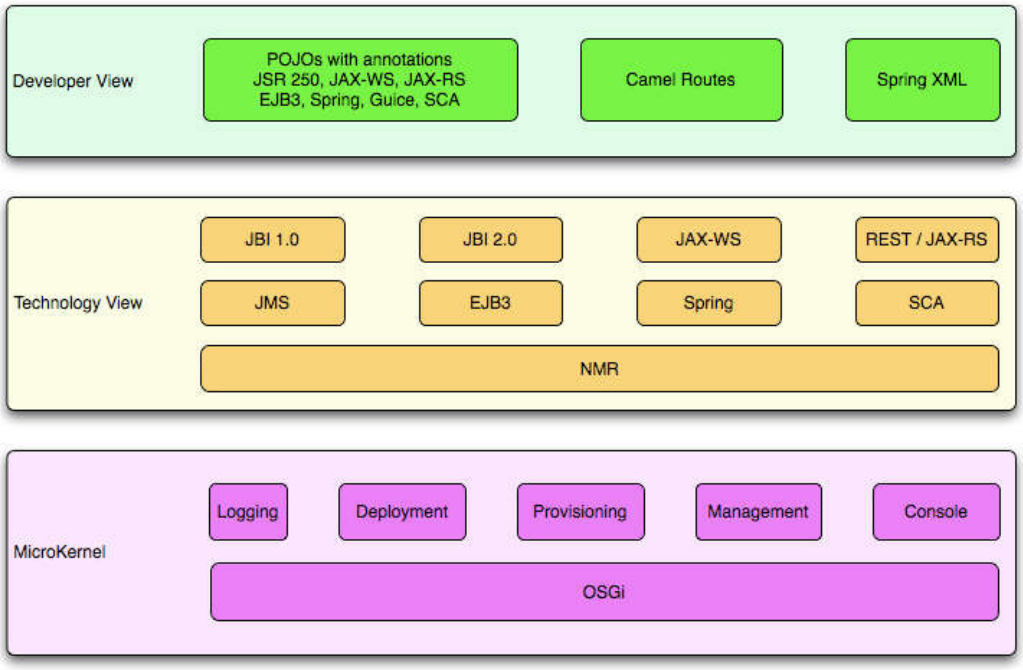


Figura 7. Arquitectura de Apache ServiceMix

Cuando se requiera la selección de una herramienta ESB para un entorno de producción se recomienda desarrollar un modelo de calidad para la evaluación del producto de acuerdo a las necesidades particulares de la organización, para el efecto pueden adoptarse las recomendaciones propuestas por ISO/IEC 25010:2011[32], este modelo deberá incluir criterios como: usabilidad, mantenibilidad, soporte, flexibilidad, funcionalidad, seguridad, disponibilidad de conectores, escalabilidad; además se deberá evaluaar: costo y tipo de licenciamiento. De acuerdo a Echeverría et al. (2008) los aspectos no funcionales son puntos clave en las soluciones de integración de fuentes de datos, por lo tanto recomienda hacer énfasis en estos aspectos al momento de seleccionar un producto ESB.

En el presente trabajo, se ha elegido la herramienta Apache Servicemix, puesto que es una plataforma lo suficientemente robusta para probar la arquitectura planteada, además que no se debe incurrir en costos por licenciamiento.

[32] https://www.iso.org/standard/35733.html

4.2 PATRONES DE INTEGRACIÓN

Las necesidades de integración de fuentes de datos son comunes en la mayoría de organizaciones y son mayores conforme crece el tamaño o la complejidad tecnológica de la organización. Históricamente, se han desarrollado distintas maneras de enfocar el problema de la integración. Desde los archivos planos, o las bases de datos relacionales, hasta los más recientes productos del tipo "*hub and spoke*" y "ESB"; la evolución en el tratamiento de la integración se ha dirigido hacia la reducción del acoplamiento entre los distintos agentes que intervienen.

Con la evolución de las tecnologías para integración de fuentes de datos, se han desarrollado patrones de integración, los cuales pretenden estandarizar las soluciones, es decir, que cuando se deban resolver diferentes problemas que requieran los mismos procedimientos, se apliquen mecanismos anteriormente utilizados y probadas en la resolución del mismo tipo de situación y así disminuir el esfuerzo en su construcción. Además los patrones de integración estandarizan la realización de ciertas tareas dentro de la integración de fuentes de datos, definen diseños comunes y determinan un lenguaje común para la comunicación entre diferentes fuentes.

Los patrones proporcionan una guía de diseño independiente de la tecnología para desarrolladores y arquitectos para describir y desarrollar soluciones de integración sólidas. Los patrones han evolucionado para volverse más específicos y solventar necesidades como: variedad en plataformas y tecnologías, estándares poco definidos, aplicaciones de terceros, socios comerciales, etc.

En muchos casos los proveedores de software proveen herramientas que proporcionan integración cruzada entre plataformas, así como la capacidad de interactuar con muchas de las aplicaciones de negocio que comúnmente tienen las organizaciones. Sin embargo, esta infraestructura técnica sólo presenta una pequeña parte de las complejidades de integración. Los verdaderos desafíos de la integración abarcan muchas cuestiones empresariales y técnicas.

Los proyectos de integración generalmente están enmarcados en las siguientes categorías:

- Portales de información
- Replicación de datos
- Funciones empresariales compartidas
- Arquitecturas Orientadas a Servicios
- Procesos de negocio distribuidos
- Integración *Business-to-Business*

Hohpe y Woolf (2003) identifican alrededor de sesenta y cinco patrones aplicados en diferentes arquitecturas de integración. A continuación se presentan algunos de los patrones analizados para el modelo de integración propuesto en este trabajo:

Content-Based Router

El patrón de integración *Content-Based Router* examina el contenido del mensaje y dirige el mensaje a un canal diferente en función de los datos contenidos en el mensaje. El enrutamiento puede basarse en una serie de criterios tales como la existencia de campos, valores de campos específicos, etc. Cuando se implementa un enrutador basado en contenido, se debe tener especial cuidado en el diseño para que la función de enrutamiento sea fácil de mantener. En escenarios de integración más sofisticados, el enrutador basado en contenido puede adoptar la forma de un motor de reglas que infiere el canal de destino sobre la base de un conjunto de reglas configurables.

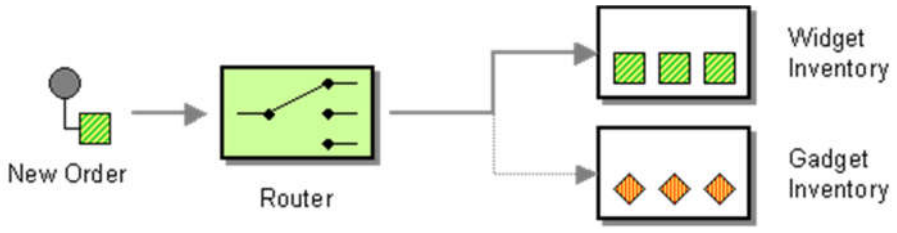


Figura 8. Patrón de Integración Content-Based Routed. (Hohpe & Woolf, 2003)

Pipes and Filters

En muchos escenarios de integración, un solo evento desencadena una secuencia de pasos de procesamiento, donde cada uno realiza una función específica. El estilo arquitectónico *Pipes and Filters* debe utilizarse para dividir una tarea de procesamiento más grande en una secuencia de pasos de procesamiento independientes más pequeños (*Filters*) conectados por canales (*Pipes*).

Cada filtro expone una interfaz muy simple: recibe un mensaje en el canal entrante, procesa el mensaje y publica los resultados en el canal saliente. El canal conecta un filtro al siguiente, enviando mensajes de salida de un filtro al otro. Debido a que todos los componentes utilizan la misma interfaz externa se pueden componer en diferentes soluciones conectando los componentes a diferentes canales. En este entorno es factible agregar nuevos filtros, omitir los existentes o reordenarlos en una nueva secuencia sin tener que cambiar los propios filtros. La conexión entre el filtro y el canal se denomina a veces puerto. En la forma básica, cada componente de filtro tiene un puerto de entrada y un puerto de salida.

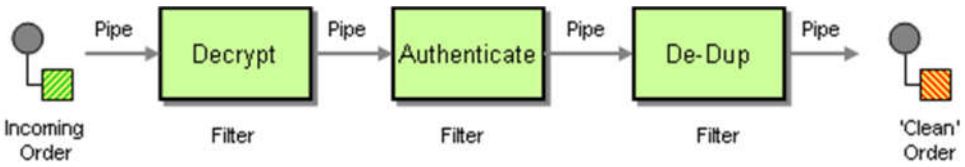


Figura 9. Patrón de Integración Pipes and Filters. (Hohpe & Woolf, 2003)

Message Router

El patrón *Message Router* difiere del concepto básico de *Pipes and Filters* debido a que se conecta a múltiples canales de salida. Gracias a la arquitectura *Pipes and Filters,* los componentes que rodean al *Message Router* no tienen conocimiento de la existencia de un *Message Router*. Una propiedad clave del patrón *Message Router* es que no modifica el contenido del mensaje. Sólo se ocupa del destino del mensaje.

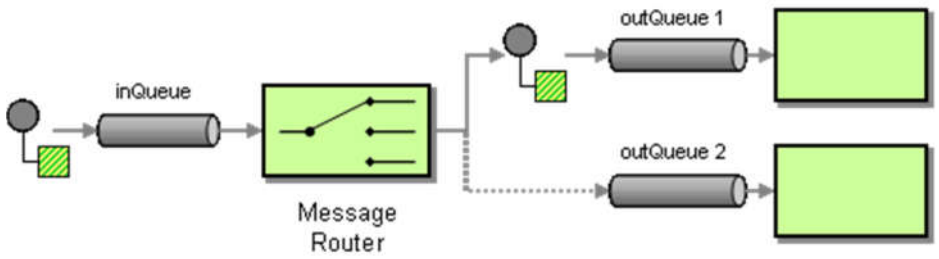


Figura 10. Patrón de Integración Message Router. (Hohpe & Woolf, 2003)

Message Filter

Este es un tipo especial del patrón *Message Router*, consiste en un filtro de mensajes, para eliminar mensajes no deseados de un canal en base a un conjunto de criterios.

El filtro de mensajes tiene un único canal de salida. Si el contenido del mensaje coincide con los criterios especificados por el filtro de mensajes, el mensaje se enruta al canal de salida. Si el contenido del mensaje no coincide con los criterios, el mensaje se descarta.

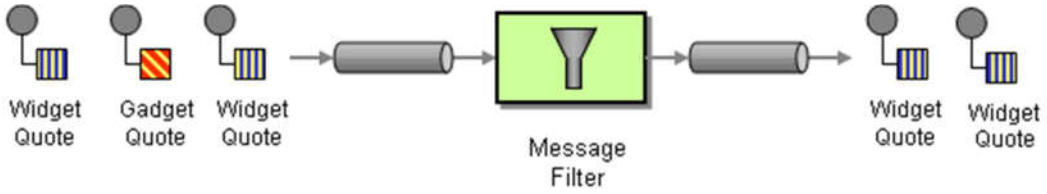


Figura 11. Patrón de Integración Message Filter. (Hohpe & Woolf, 2003)

Splitter

El patrón *Splitter* permite la división de un mensaje compuesto en una serie de mensajes individuales, cada uno de los cuales contiene datos relacionados por medio de un elemento.

Este patrón consume un mensaje que contiene una lista de elementos, cada uno de los cuales se puede procesar individualmente. El patrón *Splitter* publica un mensaje para cada elemento individual (o un subconjunto de elementos) del mensaje original.

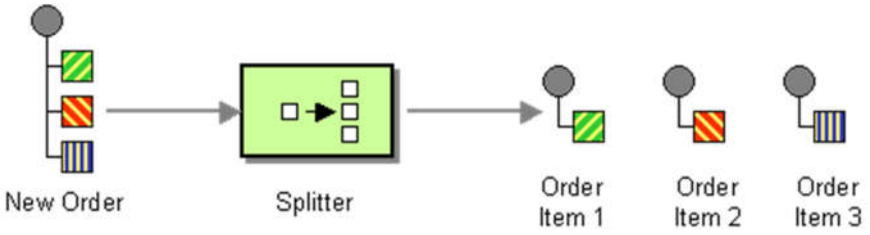


Figura 12 Patrón de Integración Splitter. (Hohpe & Woolf, 2003)

Dynamic Router

Dynamic Router es un enrutador que se puede autoconfigurar basado en mensajes de configuración especiales de los destinos participantes. Además de los habituales canales de entrada y salida, el enrutador dinámico utiliza un canal de control adicional. Durante el inicio del sistema, cada destinatario potencial envía un mensaje especial al enrutador dinámico en este canal de control, anunciando su presencia y determinando las condiciones bajo las cuales puede manejar un mensaje.

El enrutador dinámico almacena las "preferencias" para cada participante en una base de reglas. Cuando llega un mensaje, el enrutador dinámico evalúa todas las reglas y dirige el mensaje al destinatario cuyas reglas se cumplen. Esto permite un enrutamiento eficiente y predictivo sin la necesidad de mantenimiento del enrutador dinámico en cada destinatario potencial.

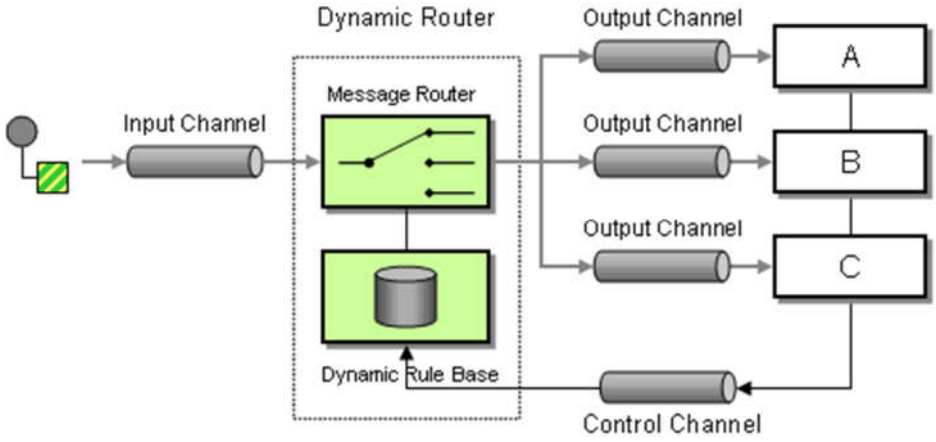


Figura 13. Patrón de Integración Dinamyc Router. (Hohpe & Woolf, 2003)

Aggregator

El patrón *Aggregator* permite recopilar y almacenar mensajes individuales hasta que se haya recibido un conjunto completo de mensajes relacionados. A continuación, el patrón *Aggregator* publica un único mensaje generado a partir de los mensajes individuales.

El diseño arquitectónico *Aggregator* es un filtro especial que recibe un flujo de mensajes e identifica los mensajes que están correlacionados para luego publicar un solo mensaje consolidado al canal de salida para su procesamiento posterior.

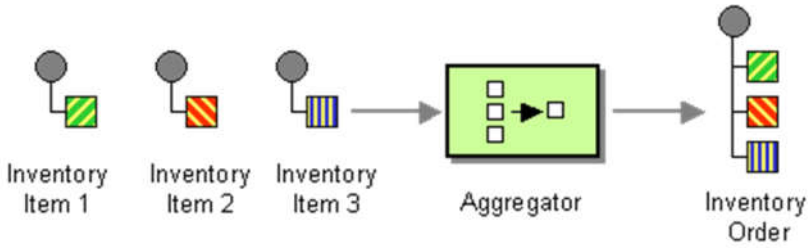


Figura 14. Patrón de Integración Aggregator. (Hohpe & Woolf, 2003)

Message Translator

Los patrones anteriores describen cómo construir mensajes y cómo encaminarlos al destino correcto. En muchos casos, las soluciones de integración empresarial enrutan mensajes entre aplicaciones existentes, como sistemas heredados, aplicaciones empaquetadas, aplicaciones personalizadas o aplicaciones manejadas por entidades externas.

Cada una de estas aplicaciones generalmente se construye alrededor de un modelo de datos propietario. El modelo de datos subyacente de la aplicación suele impulsar el diseño del esquema de la base de datos física, un formato de archivo de interfaz o una interfaz de programación, las entidades con las que una solución de integración tiene que interactuar. Como resultado, las aplicaciones esperan recibir mensajes que imitan el formato de datos interno de la aplicación.

Además de los modelos de datos propietarios y los formatos de datos incorporados en las diversas aplicaciones, las soluciones de integración muchas veces interactúan con formatos de datos estandarizados que buscan ser independientes de aplicaciones específicas. Existen varios consorcios y organismos de normalización que definen estos protocolos, como ebXML[33], OAGIS[34] y muchos otras organizaciones en la industria. En muchos casos, la solución de integración debe poder comunicarse con partes externas utilizando los formatos de datos "oficiales", mientras que los sistemas internos se basan en formatos propietarios. *Message Translator* convierte un mensaje de un componente en otro mensaje para que pueda utilizarse en un contexto diferente.

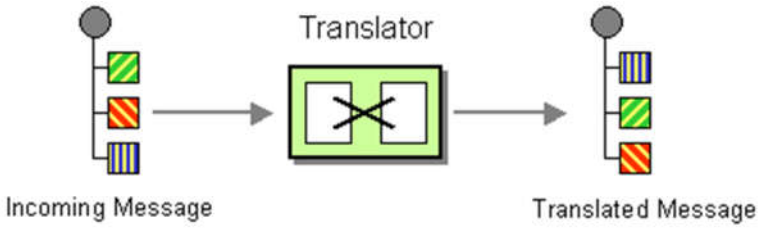


Figura 15. Patrón de Integración Message Translator. (Hohpe & Woolf, 2003)

Message Endpoint

Las fuentes de datos se comunican enviando mensajes entre sí a través de canales de mensajes. Para la utilización de patrón *Message Endpoint* el código debe ser personalizado tanto para la aplicación como para la interface de programación del cliente. El resto de la aplicación conoce poco sobre los formatos de mensajes, canales de mensajería, o cualquiera de los otros detalles de la comunicación con otras aplicaciones. Sólo conoce que tiene una solicitud o elemento de datos para enviar a otra aplicación, o está esperando un mensaje de otra aplicación.

[33] http://www.ebxml.org/
[34] http://www.oagi.org/dnn2/

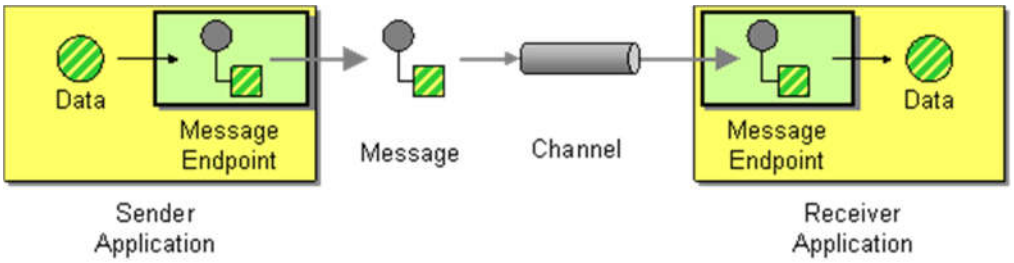


Figura 16. Patrón de Integración Message Endpoint. (Hohpe & Woolf, 2003)

Publish-Subscribe Channel

La notificación de eventos entre aplicaciones se realiza por medio de mensajería para el efecto el patrón *Publish-Subscribe Channel* dispone de un canal de entrada que se divide en múltiples canales de salida, uno para cada abonado. Cuando se publica un evento en el canal, el canal de publicación entrega una copia del mensaje a cada uno de los canales de salida. Cada canal de salida tiene solamente un abonado, que permite consumir un mensaje a la vez. De esta manera, cada abonado sólo obtiene el mensaje una vez y las copias consumidas desaparecen de sus canales.

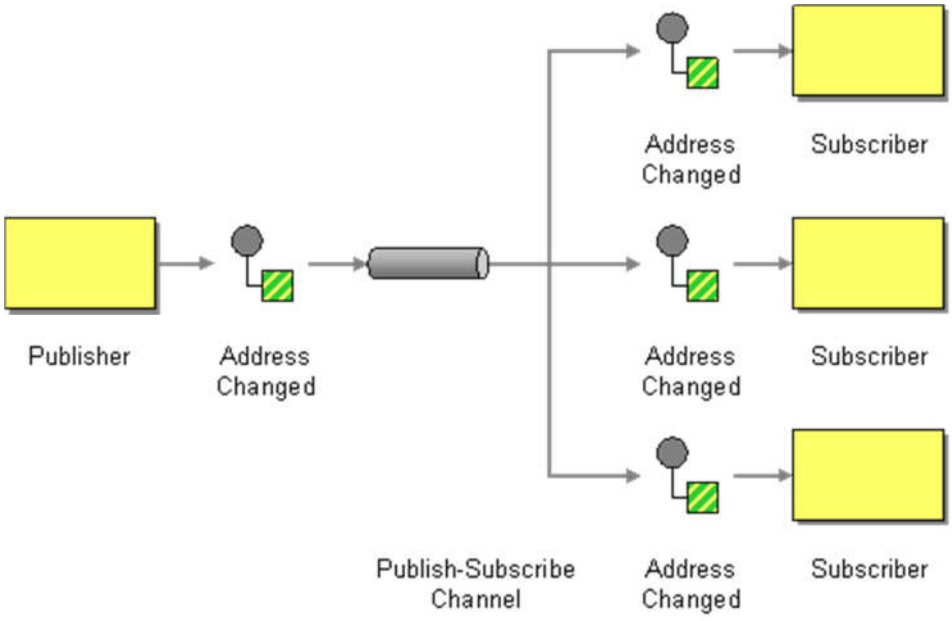


Figura 17. Patrón de Integración Publish-Subscribe Channel. (Hohpe & Woolf, 2003)

Event-Driven Consumer

Una aplicación requiere consumir mensajes tan pronto como se entregan. La aplicación debe utilizar un consumidor impulsado por eventos que entrega automáticamente los mensajes a medida que se colocan en el canal.

Esto también se conoce como receptor asíncrono, porque el receptor no tiene un subproceso en ejecución hasta que un hilo de retorno de llamada envíe un mensaje, se denomina un consumidor impulsado por eventos porque el receptor actúa como si la entrega de mensajes fuera un evento que activará al receptor en acción.

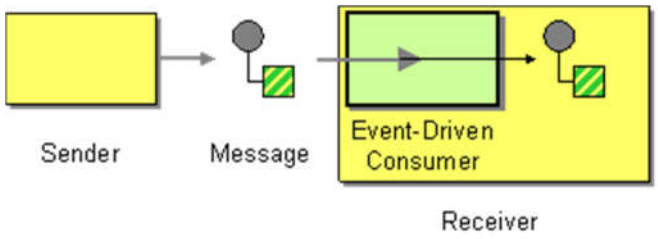


Figura 18. Patrón de Integración Event-Driven Consumer. (Hohpe & Woolf, 2003)

Composed Message Processor

El patrón *Composed Message Processor* divide el mensaje, dirige los sub-mensajes a los destinos apropiados y vuelve a agrupar las respuestas en un único mensaje.

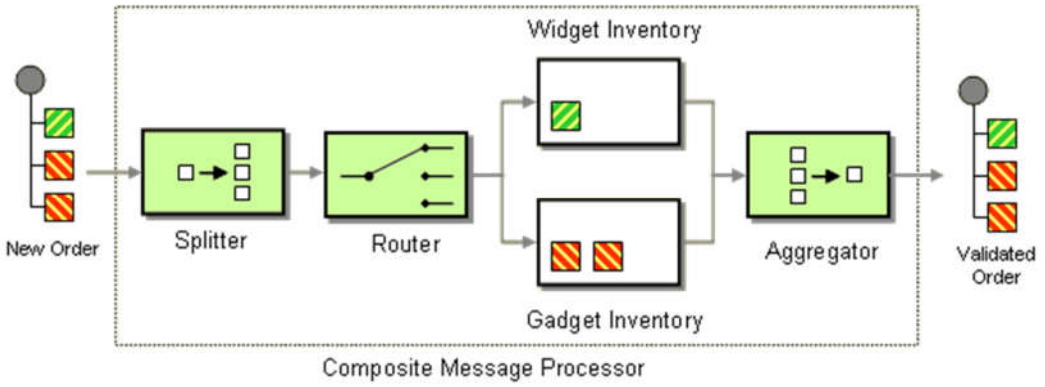


Figura 19. Patrón de Integración Composed Message Processor. (Hohpe & Woolf, 2003)

4.3 ONTOLOGÍAS

El uso del término “ontología” tiene diversos significados dependiendo de la disciplina o ámbito dentro del que se lo utilice. En el contexto de la Web y los sistemas computacionales, “una Ontología es una especificación formal y explícita de una conceptualización compartida” (Gruber, 1993). Una ontología permite especificar formalmente conceptos generales sobre un dominio, además permite definir términos y las relaciones básicas entre ellos, y por otra parte, hace posible que la información se encuentre en formatos legibles por agentes de software para su gestión. Las ontologías son acuerdos, en un contexto social, para cubrir una serie de objetivos, como permitir el intercambio de datos entre programas, simplificar la unificación (o traducción) de distintas representaciones, facilitar la comunicación entre personas.

Según Gruber (1993) las ontologías tienen los siguientes componentes que servirán para representar el conocimiento de algún dominio:

- Conceptos: son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.
- Relaciones: representan la interacción y enlace entre los conceptos del dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a, etc.
- Funciones: son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden parecer funciones como categorizar-clase, asignar fecha, etc.
- Instancias: se utilizan para representar objetos determinados de un concepto.
- Axiomas: son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: “Si A y B son de la clase C, entonces A no es subclase de B”, “Para todo A que cumpla la condición C1, A es B”, etc.

Las ontologías proporcionan una comprensión común de un dominio, de modo que se eliminan confusiones conceptuales y terminológicas. En la práctica, el desarrollo de una ontología incluye:

- Definir clases en la ontología.
- Colocar las clases en una jerarquía de taxonomías (subclase - superclase).
- Definir propiedades e instancias.
- Rellenar los valores de las propiedades con ejemplos.

Metodología NeOn

Para el tratamiento de las ontologías se han desarrollado algunas iniciativas entre ellas la Metodología NeOn[35] que surgió en el año 2006 con la participación de profesionales, en los campos de la ingeniería ontológica, tecnologías de colaboración, ingeniería del software e interacción humano-computador. El objetivo principal del proyecto fue mejorar la capacidad para manipular múltiples ontologías en red, incluyendo los cambios y mantenimiento, además de guiar en el proceso de generación de ontologías colaborativas y la reutilización de recursos ontológicos y no ontológicos.

A continuación se presentan los escenarios para el desarrollo de ontologías propuesto por Suárez y Figueroa (2010):

- Escenario 1: Desde la especificación de la aplicación: para iniciar la construcción de una ontología es necesario especificar los requisitos; luego, desarrollar la búsqueda de los posibles recursos para reutilizar; y, proceder con la planificación del trabajo. Para ello, se consideran aspectos como las necesidades, intenciones, usos, usuarios, entre otros. Como resultado de este escenario se obtiene el Documento de Especificación de Requisitos de la Ontología
- Escenario 2: La reutilización y reingeniería de los recursos no ontológicos (NOR): en este escenario los desarrolladores deben ejecutar el proceso de reutilización de NORs, determinando los NORs a reutilizar para realizar

[35] http://www.neon-project.org/

la reingeniería. Estos recursos son fuentes de conocimiento por lo general heterogéneos cuya semántica no se ha establecido en una ontología, como por ejemplo, glosarios, vocabularios, diccionario de sinónimos, entre otros.

- Escenario 3: La reutilización de los recursos ontológicos: se determinan los recursos ontológicos o declaraciones ontológicas que se utilizarán en la red de ontologías.
- Escenario 4: La reutilización y re-ingeniería de los recursos ontológicos: consiste en reutilizar y reorganizar los recursos ontológicos. NeOn incluye una librería de patrones para el diseño de ontologías. Los patrones son de tipo estructural, correspondencia, razonamiento, léxico-sintaxis y contenido.
- Escenario 5: La reutilización y la fusión de los recursos ontológicos: en este escenario se realiza la unión de los recursos que se pueden reutilizar con nuevos recursos que se requiere crear.
- Escenario 6: Reutilización, la fusión y re-ingeniería de los recursos ontológicos: se realiza la reingeniería de los recursos ontológicos combinados, similar al escenario 5.
- Escenario 7: Reutilización de los patrones de diseño de ontologías (ODPs): se recomienda recurrir a los patrones de diseño establecidos para solucionar problemas recurrentes en el proceso de diseño de ontologías.
- Escenario 8: Reestructuración de recursos ontológicos: los recursos ontológicos se reestructuran (modularización, poda, extensión y/o especialización) para luego adaptarlos a la red de ontologías.
- Escenario 9: Localización de recursos ontológicos: se adapta la ontología a otras lenguas y culturas, con el fin de obtener una ontología multilingüe.

En la Figura 20 se visualizan los escenarios de la metodología NeOn descritos anteriormente.

La metodología NeOn también incluye un glosario de procesos y actividades que participan en la construcción de redes de ontologías, producto de un consenso

entre ingenieros, editores y usuarios. Además, la metodología NeOn plantea dos modelos para organizar los procesos y actividades que se ejecutan durante el desarrollo de una ontología: Cascada e Incremental-iterativo.

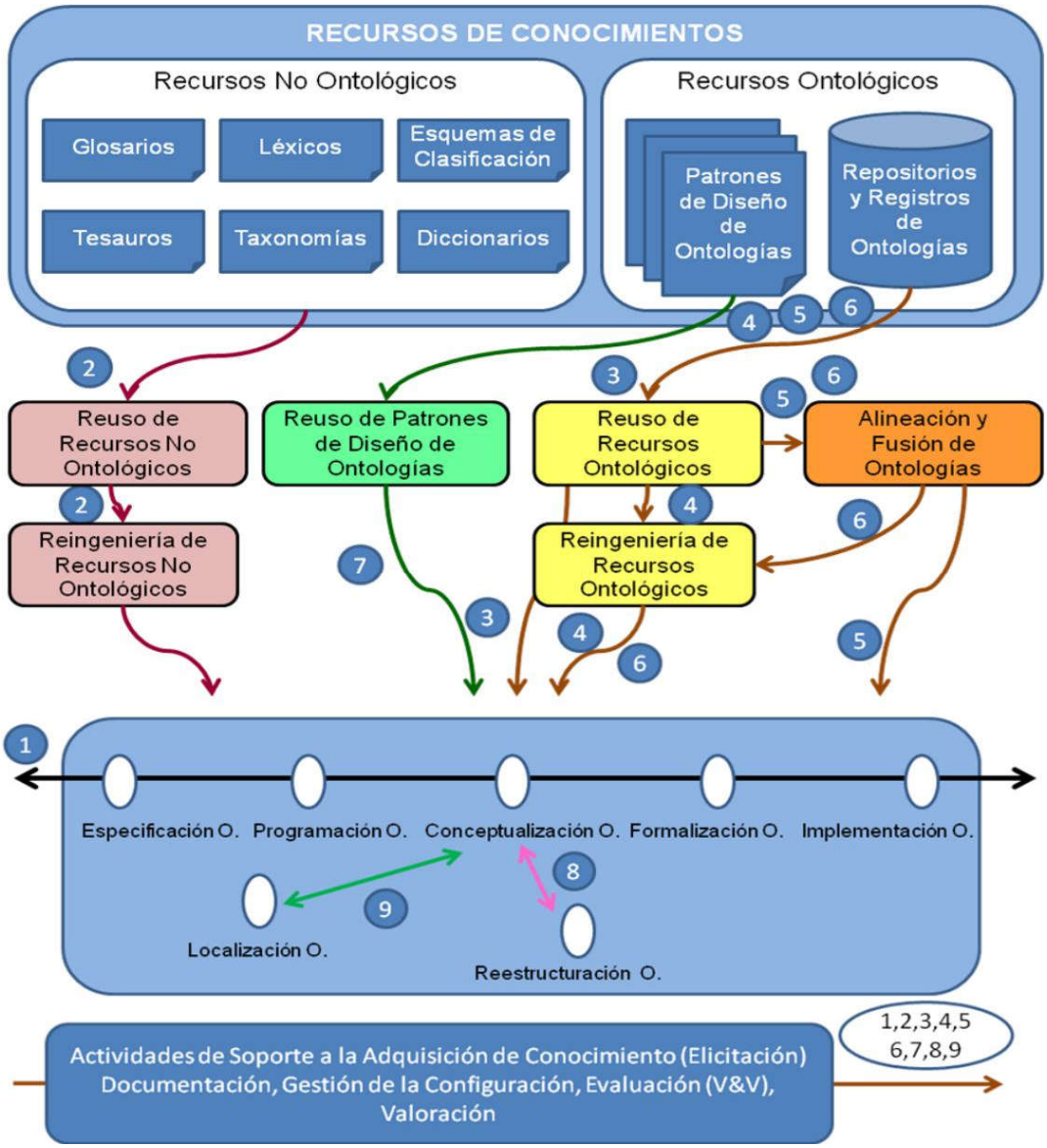


Figura 20. Escenarios de la Metodología NeOn (Suárez y Figueroa,2010)

Por otra parte, la metodología NeOn propone un conjunto de guías que describen como ejecutar los diferentes procesos. La metodología NeOn no es un marco de trabajo rígido, por el contrario contempla algunas alternativas para el desarrollo de redes de ontologías, las características enunciadas anteriormente hacen que

NeOn sea la metodología recomendada como soporte en la construcción de ontologías dentro de la arquitectura de integración que se plantea en este documento.

5. ESCENARIO

Para la elaboración de la propuesta se realizó el análisis de las fuentes de datos existentes en la Universidad de Cuenca que en la actualidad cuenta con dieciocho fuentes de datos, de donde se extrajo un caso típico que generalmente se repite en otras organizaciones, como es la gestión de recursos humanos. Para fines demostrativos de los problemas de integración se construyeron dos aplicaciones prototipo que gestionan la entidad “persona” la cual se emplea en varias fuentes de datos en la organización analizada.

El modelo seguido para la implementación de los prototipos es el Modelo Vista Controlador (MVC), el cual es un patrón para el desarrollo del software que se basa en separar los datos, la interfaz del usuario y la lógica interna. MVC es comúnmente usado en aplicaciones Web, donde la vista es la página HTML, el modelo es el Sistema de Gestión de Base de Datos y la lógica interna, y el controlador es el responsable de recibir los eventos y dar solución. Adicionalmente se crea una capa que contiene los servicios Web que permitirán el acceso a los datos de las diferentes aplicaciones como se muestra en la Figura 21; la descripción que se presenta a continuación es válida para las dos aplicaciones (prototipos) que se han creado y serán tomadas en consideración en el proceso de integración:

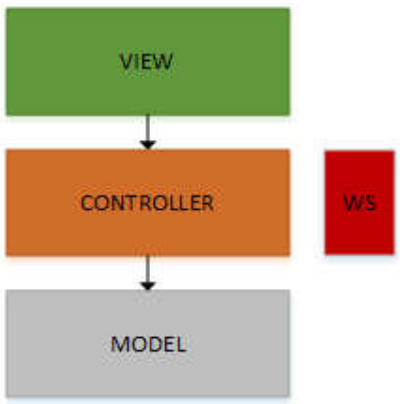


Figura 21. MVC: Arquitectura del Prototipo

5.1 VISTA

El objetivo de la interface Web implementada es soportar el ingreso de datos en el prototipo. En este caso los datos corresponden a una persona y su información de contacto. Para intentar que el prototipo refleje problemas reales de integración en las organizaciones se utilizan diferentes tecnologías en la implementación de los prototipos, entre ellas están IDEs de desarrollo, base de datos, plataformas de servidores, etc. En la creación de la interfaz de usuario se utilizaron tecnologías que permiten simular aplicaciones de las organizaciones, entre ellas está Genexus X (evolution 3)[36] como tecnología de desarrollo de aplicaciones Web (véase Figura 22).

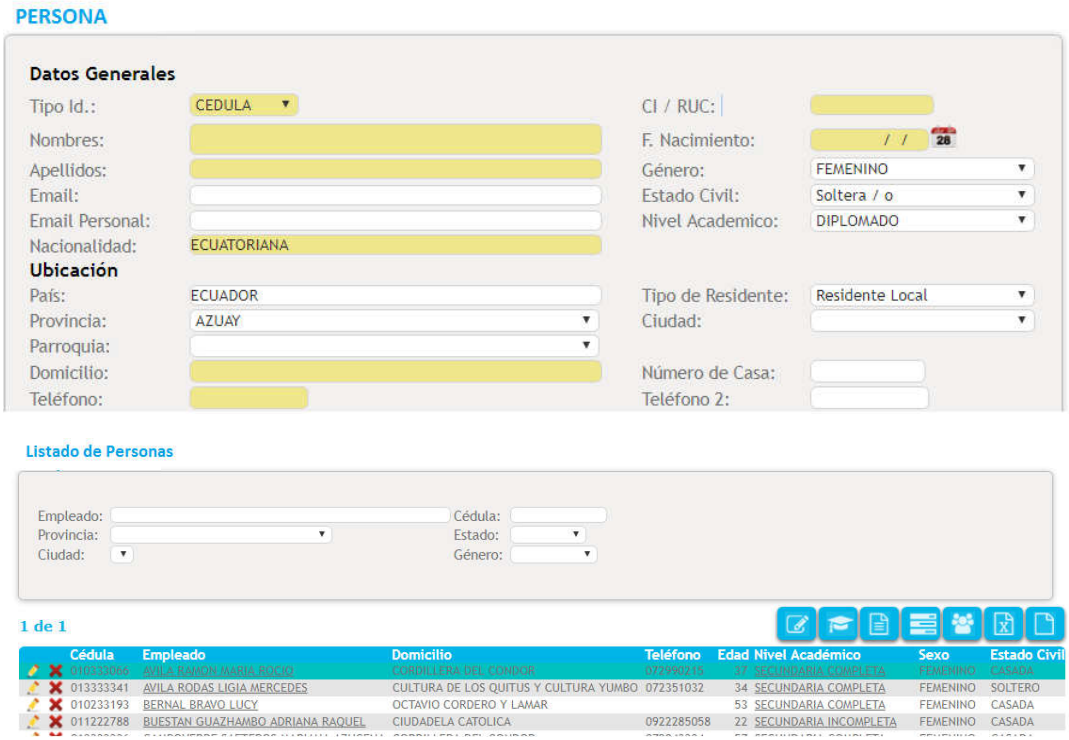


Figura 22. Interfaces para entrada y listado de Datos - Prototipo

5.2 CAPA DE SERVICIOS WEB

Este enfoque asume un escenario controlado, es decir, las fuentes de datos a integrar son propias de las organizaciones y por lo tanto son manipulables tanto a nivel de datos como de código. Es la capa que actúa junto al controlador de aplicación y para cumplir sus propósitos se han implementado servicios Web que reciben las peticiones del usuario y registran la información correspondiente en

[36] https://www.genexus.com/genexus-versiones/genexus-x-evolution-3?es

el modelo de datos. Los servicios implementados permiten ejecutar los métodos necesarios para las operaciones de ingreso, actualización, y listado de los objetos. Para el caso de este prototipo se han definido métodos que permiten tratar con la capa de datos, de esta manera se pueden gestionar los datos a partir de la fuente. Los métodos definidos son los siguientes: *create, edit, destroy, changeList* y *updateList*. (Véase Figura 23).

```
@WebService(serviceName = "PersonaWebServices")
@Stateless()
public class PersonaWebServices {

    private MPersonaJPAController controller =new MPersonaJPAController();

    @WebMethod()
    public void create(MPersona object) throws Exception {
        controller.create(object);
    }
     @WebMethod()
    public void edit(MPersona object) throws Exception {
        controller.edit(object);
    }
    public void destroy(MPersona object) throws Exception {
        controller.destroy(object);
    }
    public  List<MPersona> changeList() {
        List<MPersona> l=controller.changeList();
        return l;
    }

    public void updateList(List<MPersona> lista){
        controller.updateList(lista);
    }
}
```

Figura 23. Operaciones definidas en Servicio WEB

Para la implementación de los servicios Web se han usado estándares como: WSDL (Web Services Description Language)[37], está basada en XML[38] y permite describir la interfaz pública a los servicios Web, así como la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo; SOAP (Simple Object Access Protocol)[39] : Es un protocolo estándar que define

[37] https://www.w3.org/TR/wsdl
[38] https://www.w3.org/XML/
[39] https://www.w3.org/TR/soap/

cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

5.3 CAPA DE LÓGICA DE NEGOCIO

Esta capa es el núcleo de la aplicación. El propósito de esta capa en el prototipo es que la lógica y reglas de negocio se realicen a este nivel y no se mezclen con objetos de las otras capas. A continuación describimos los diferentes métodos que interactúan con el servicio Web descrito en la capa anterior:

- ***Create*:** método que permite la inserción de los datos, este método es consumido con el método *create* del servicio Web.
- ***Update*:** método que permite la actualización de los datos, este método es consumido con el método *edit* del servicio Web.
- ***Destroy*:** método que permite la eliminación de los datos, este método es consumido con el método *destroy* del servicio Web.
- ***ChangeList*:** método que permite la obtención de los cambios de los datos, este método es consumido con el método *changeList* del servicio Web.
- ***UpdateList*:** método que permite la actualización de la variable de acción de los datos, este método es consumido con el método *updateList* del servicio Web.

5.4 MODELO DE DATOS

Para la definición del modelo de datos se analizó un fragmento de la fuente que gestiona recursos humanos que se visualiza en la Figura 24, en la cual se detallan algunas entidades y relaciones que se emplean en el sistema en producción. De este análisis y para efectos demostrativos se decidió realizar una abstracción del problema y se crearon modelos de datos sobre los cuales se describirán los problemas de integración que se tratan en este trabajo.

Los datos representan a una persona con diferentes características, es decir, cada prototipo tiene su propio modelo acerca de una persona, en el primer caso se manejan tres campos (id, nombres, apellidos) y en el segundo caso cinco campos (id, nombres, apellidos, fecha de nacimiento, dirección).

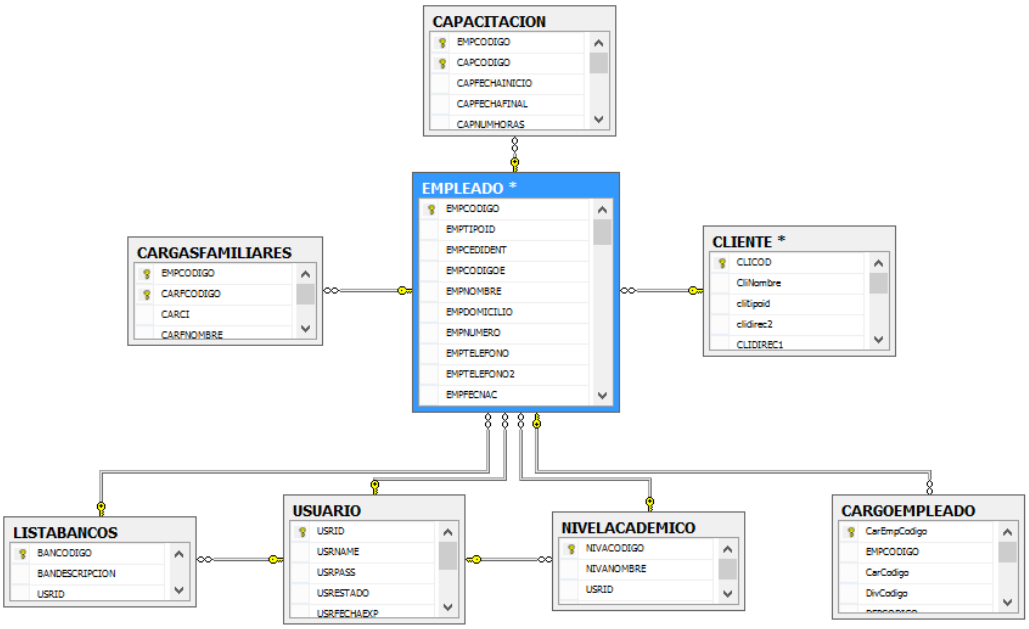


Figura 24. Diagrama de Estructura de Datos, fuente analizada

6. PROPUESTAS DE INTEGRACIÓN

En esta sección se presenta las propuestas de integración con diferentes enfoques que permitirá un acceso uniforme a los datos de la organización.

6.1 INTEGRACIÓN UTILIZANDO UN BUS DE SERVICIOS

Esta solución hace uso de un ESB mediante la aplicación del patrón EAI (Enterprise Application Integration) a una arquitectura orientada a servicios, Linthicum (2000). Una solución EAI consiste en comunicar las diferentes aplicaciones mediante conectores, tanto dentro de la organización como fuera de ella.

Una de las ventajas de usar un ESB, es que este posibilita la comunicación entre fuentes de datos independiente del protocolo usado. Es decir, se convierte en una pasarela que se encarga de traducir de un lenguaje a otro. El lenguaje usado en el ESB es XML, lo cual facilita el intercambio de mensajes.

En esta propuesta, los servicios web no interactúan directamente, éstos deben ser registrados en el ESB, y su invocación se realiza a través de un conector de servicios SOAP. Internamente en el ESB se definen rutas utilizando diferentes patrones de integración que permiten definir un modelo virtual de integración. Es necesario señalar, que cuando se requiera integrar una nueva aplicación, se tienen que crear los servicios web correspondientes para registrarlos dentro del ESB. Si en lugar de una nueva aplicación es una base de datos (BD), se debe registrar la BD en el ESB y utilizar un componente de acceso exclusivo para este tipo de fuentes.

En ambos casos, es primordial el conocimiento acerca del modelo de datos de cada fuente, puesto que las rutas que se definen en el ESB se realizan en base a las relaciones que puedan existir entre las entidades de las fuentes de datos.

6.2 INTEGRACIÓN UTILIZANDO UN BUS DE SERVICIOS MÁS TECNOLOGÍAS SEMÁNTICAS

En esta subsección se introduce el concepto de acceso uniforme de datos usado para la integración de fuentes de datos aplicando herramientas basadas en tecnologías semánticas dentro de un ESB. Uno de los factores considerados para optar por una propuesta semántica de integración es que esta solución permite afrontar uno de los retos claves en la integración de datos de distintas fuentes, la heterogeneidad semántica (introducida en la Sección 3.1). Además, este enfoque permitirá definir modelos de integración tanto virtuales como materializados.

Este planteamiento tiene la limitante de ser exclusivamente virtual, es decir, no se mantiene un repositorio común de datos. Si una aplicación esta fuera de línea, puede ocasionar que las rutas definidas que permiten tratar con las fuentes de datos no se ejecuten, razón por la se plantea adicionalmente que el modelo posibilite el funcionamiento en línea como fuera de línea, para el efecto se contempla la creación de un almacén de datos semántico centralizado. A continuación se describen los componentes principales del prototipo semántico implementado.

6.2.1 ONTOLOGÍAS COMO SOLUCIÓN A LA INTEGRACIÓN SEMÁNTICA

Se ha adoptado la arquitectura de referencia ANSI/SPARC[40] de tres capas (esquema físico, esquema conceptual y vistas) para que soporte semántica. En esta versión adaptada, se establecen tres capas análogas: documentos, esquema y ontología (véase Figura 25). Esta división en capas permite separar y solucionar los problemas existentes en todo el proceso de la integración semántica.

[40] https://www.info.teradata.com/HTMLPubs/DB_TTU_16_00/index.html#page/Database_Management/B035-1094-160K/tpt1472240595833.html

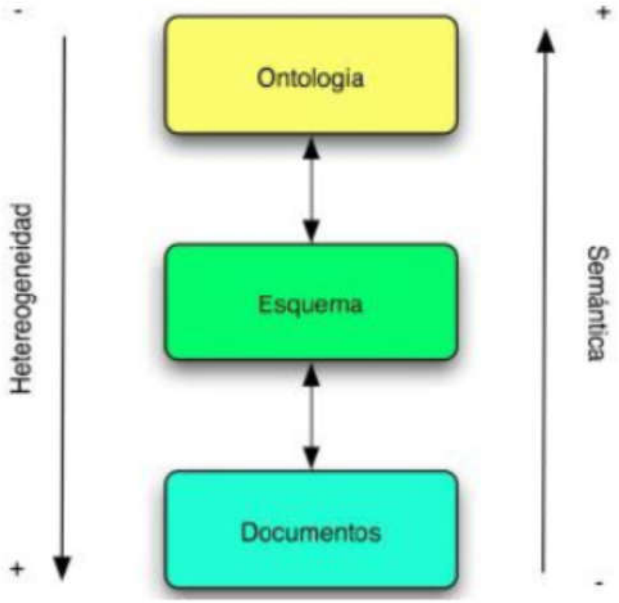


Figura 25. Capas para un proyecto de integración semántica

- Documentos: La capa de documentos contiene los datos que se pretende integrar, en su formato nativo o bien se puede utilizar un recubridor que efectué la conversión necesaria hacia documentos XML, RDF u OWL.
- Esquema: La capa de esquema describe la estructura global de los documentos que componen la capa inferior. Se tratará de descripciones mediante XML Schema, RDF u OWL.
- Ontología: La capa de ontología proporciona una visión semánticamente coherente de la información mediante el uso de ontologías que describan el dominio del sistema. El usuario siempre interactúa con esta capa, que proporciona una visión simplificada y de alto nivel, ocultando la heterogeneidad del sistema subyacente. En este caso la Ontología se construye bajo el lenguaje de ontologías OWL.

En la literatura revisada se encontraron varios autores, quienes afirman que las ontologías aparecen como solución potencial para resolver el problema de la integración semántica de los datos. Esta propuesta ha sido implementada en el prototipo y los detalles del mismo se explican a continuación.

6.2.2 ARQUITECTURA DEL PROTOTIPO

La plataforma de integración que se propone debe ser fácil de usar, administrar e integrar en un entorno existente. La solución abarca una infraestructura semánticamente enriquecida orientada a servicios, que incluye un bus de servicios empresarial en este caso semántico para el control, manejo y transformación de los mensajes.

La arquitectura del prototipo propuesto fue diseñada de acuerdo con los principios de SOA como altamente modular y extensible. El proceso de creación de la plataforma semántica se compone de las siguientes etapas:

1) Metodología de creación de ontologías:

Durante esta etapa se usó la Metodología NeOn, la cual permite la creación de recursos ontológicos partiendo de vocabularios existentes y similares al dominio de aplicación. Se debe destacar que como primer paso se busca recursos similares accediendo a repositorios disponibles en la Web. En caso que estos recursos no sean adecuados para los propósitos del modelo a implementar, se procede a la etapa dos de la metodología. Esta segunda etapa propone buscar recursos no ontológicos como bases de datos, catálogos, etc., para la construcción de la ontología. Para el escenario descrito se ha definido un modelo ontológico para cada prototipo, modelo que representa a cada fuente de datos. Además, se define un modelo ontológico global que permita mantener una visión homogénea y unificada de toda la organización, este modelo global debe estar basado en la estructura organizacional de la empresa.

2) Gestión de ontologías

Una vez definidas las ontologías, durante esta etapa se procede a revisar los conceptos y propiedades que conforman cada prototipo de forma que permitan el intercambio adecuado de conocimiento. En este caso se crearon componentes dentro del ESB que permiten manipular las ontologías, es decir, realizar operaciones de inserción, actualización, borrado y listado de entidades. Estos componentes creados tienen una estrecha relación con los servicios web

descritos anteriormente, puesto que parte de los modelos ontológicos circulan a través de las rutas creadas dentro del ESB.

3) Relación entre ontologías

Finalizada la definición de los conceptos y propiedades de cada ontología (locales y global), se procede a relacionar los recursos ontológicos, es decir, se genera los mapeos entre las entidades del modelo global y los modelos locales. La generación de estos componentes se realizó con la aplicación Protegé[41], que es una herramienta de código abierto que permite la edición de ontologías y el marco base de conocimiento.

4) Definición de las instancias

A fin de poder hacer uso de los recursos ontológicos creados, en este paso se procede a crear las instancias para cada concepto. Hay que mencionar que los conceptos dentro de una ontología son conocidos como TBox, mientras que las instancias son conocidas como ABox, como se muestra en la Figura 26.

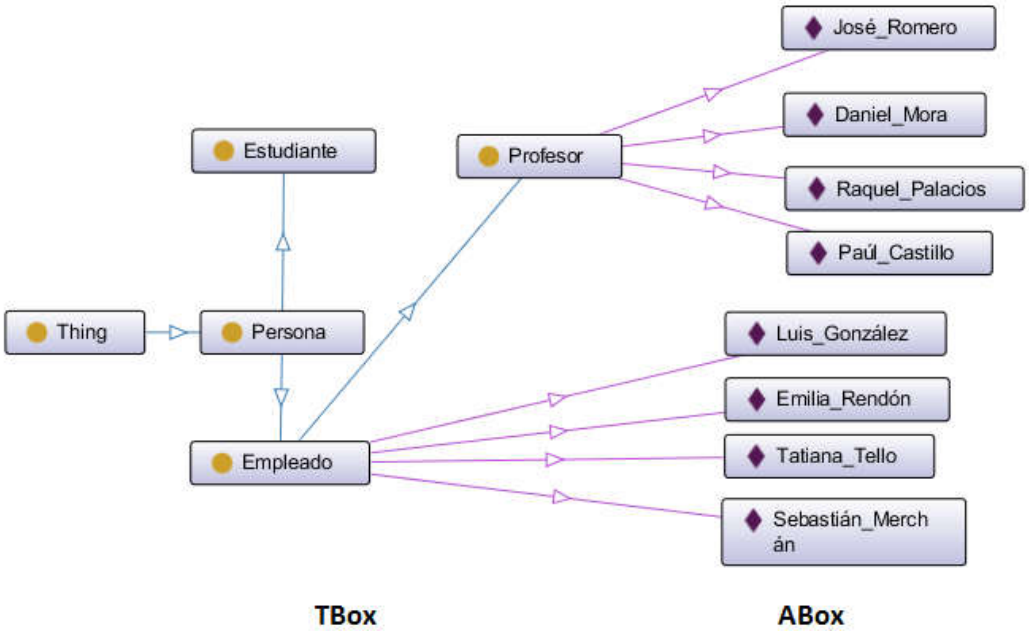


Figura 26. Definición de Instancias

[41] https://protege.stanford.edu/

5) Definición de Rutas

Las rutas definidas en este enfoque permiten leer información de los servicios Web creados en los prototipos, esta información es transformada y transportada a diferentes componentes formando un flujo de información por el cual circulan los datos de diferentes aplicaciones (véase Figura 27). Estas rutas están basadas en los patrones de integración citados en la Sección 4.2 los cuales permiten mantener un estándar de extracción, transformación de los datos de los servicios Web y carga en el repositorio semántico.

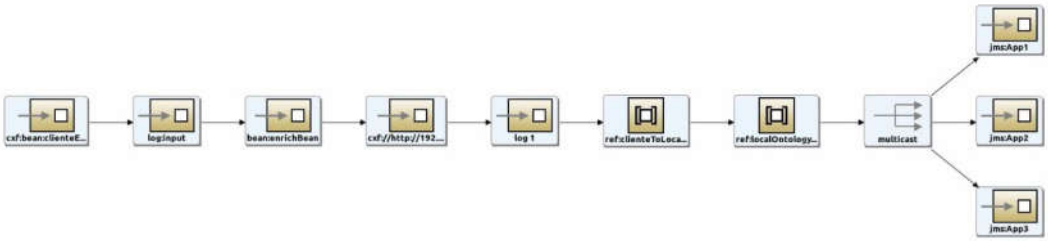


Figura 27. Definición de Rutas

6.2.3 MODELO DE COMUNICACIÓN

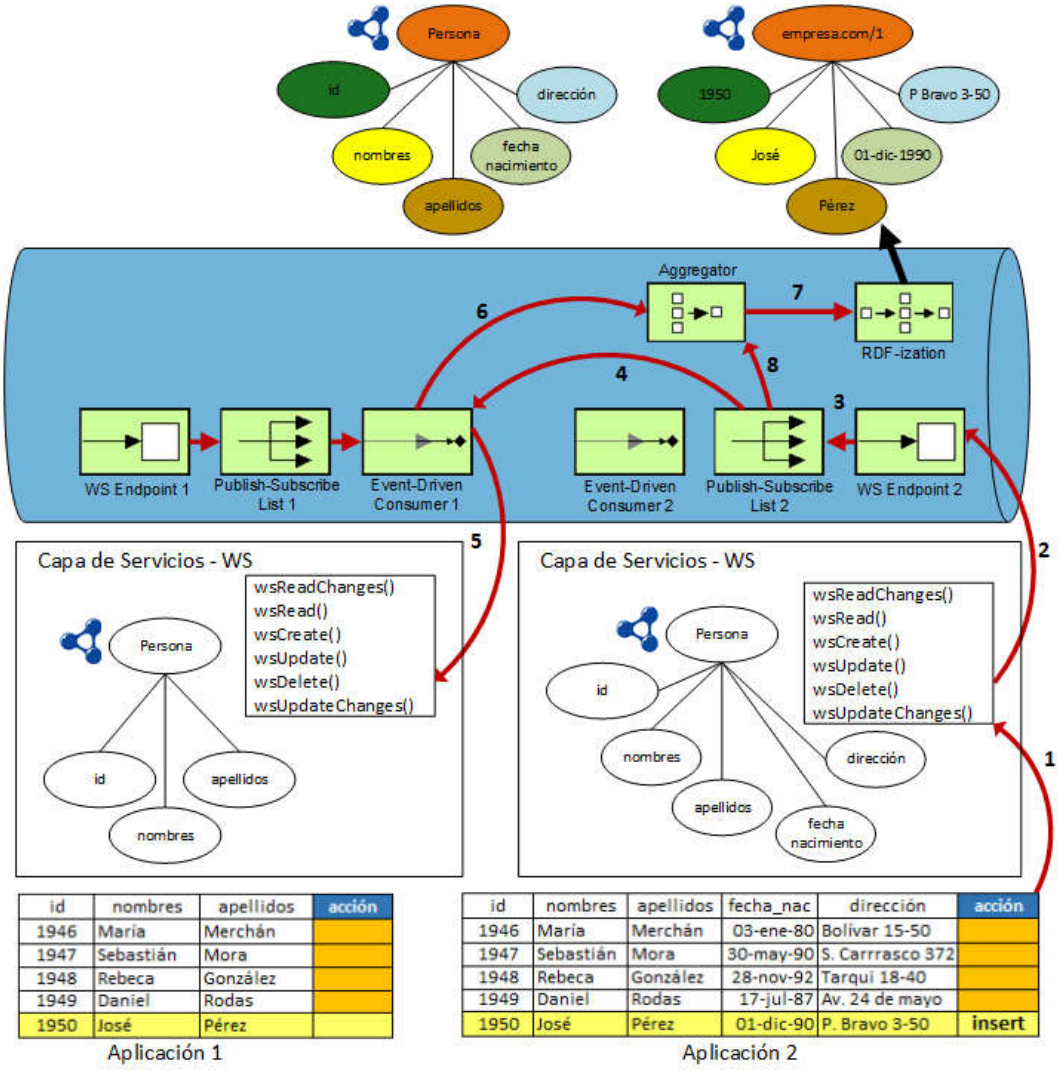
En la Figura 28 se muestra el proceso completo de inserción/actualización de información utilizando un ESB, así como los diferentes componentes que interactúan. Para clarificar el ejemplo, se ha definido un problema de integración con los dos prototipos definidos en el escenario e implementados utilizando MVC. Se asume que se tiene la entidad persona en los dos prototipos, en el primer caso existen 3 campos (id, nombres, apellidos) y en el segundo caso 5 campos (id, nombres, apellidos, fecha de nacimiento, dirección). Como modelo global se tiene un modelo ontológico con los conceptos (id, nombres, apellidos, fecha de nacimiento, dirección).

1) El proceso inicia con la inserción de un nuevo registro en la aplicación dos (1950,'José','Pérez','01/04/1990','P. Bravo 3-50', insert), puesto que las fuentes de datos son controladas, se ha creado un campo más a nivel de la BD (acción), la que permite conocer la acción a realizar sobre el registro. En este caso indica inserción de datos. En la capa de servicios Web de la aplicación dos existen diferentes métodos que permiten manipular los registros de la base de datos.

2) En el ESB existe un componente (WS Endpoint 2) que cada cierto tiempo accede a los servicios Web de la aplicación dos en busca de registros nuevos,

actualizados o borrados. En este caso detecta que existe un registro nuevo (1950,'José','Pérez','01/04/1990','P. Bravo 3-50', insert) que fue detectado a través del método *wsReadChanges*.

3) Los datos extraídos a través de los servicios Web son depositados en un componente basado en *publish-subscribe* el cual permite mantener los datos y que otros componentes accedan a ellos.



id	nombres	apellidos	acción
1946	María	Merchán	
1947	Sebastián	Mora	
1948	Rebeca	González	
1949	Daniel	Rodas	
1950	José	Pérez	

id	nombres	apellidos	fecha_nac	dirección	acción
1946	María	Merchán	03-ene-80	Bolívar 15-50	
1947	Sebastián	Mora	30-may-90	S. Carrrasco 372	
1948	Rebeca	González	28-nov-92	Tarqui 18-40	
1949	Daniel	Rodas	17-jul-87	Av. 24 de mayo	
1950	José	Pérez	01-dic-90	P. Bravo 3-50	insert

Figura 28. Modelo de Integración Aplicando Tecnologías Semánticas

4) De la misma manera que la aplicación dos, la aplicación uno dispone de una capa de servicios Web para la manipulación de los registros en la base de datos. En este caso, puesto que la operación a realizar es una inserción en la aplicación uno, un componente registrado para la aplicación uno está pendiente si existen

cambios en el repositorio común (*publish-subscribe*). En este ejemplo detecta un registro que viene de la aplicación dos y registra en su repositorio local.

5) El nuevo registro se envía a través de los servicios web de la aplicación uno (*wsCreate*), esto permite almacenar el registro creado en la aplicación dos en la base de datos de la aplicación uno.

6) Simultáneamente con el envío de los datos a la aplicación uno, un componente envía el registro del repositorio local de la aplicación uno a un componente temporal de agregación (*Aggregator*).

7) En este paso, los registros alojados en el componente temporal de agregación son transformados a RDF utilizando el modelo global (*RDF-ization*). Permitiendo mantener una base de datos semántica de todos los registros de la organización.

8) Simultáneamente con el registro de los datos en el componente *Publish-Subscribe List 2*, se envía el registro a un componente de agregación temporal.

El proceso de creación/actualización/borrado de registros en las diferentes fuentes de datos se gestiona de la misma manera, con ligeras variaciones en el flujo de datos. Como se puede observar el proceso es generalista, permitiendo la agregación de nuevas fuentes de datos de forma prácticamente instantánea. El proceso más complejo es la detección de modelo ontológico local de la nueva aplicación a integrar, puesto que los componentes son generales para cualquier aplicación.

6.2.4 IMPLEMENTACIÓN DE COMPONENTES PARA MANEJO DE ONTOLOGÍAS

Dentro de la propuesta planteada fue necesaria la construcción de componentes para la manipulación de recursos ontológicos, para el efecto se utilizó el framework Jena[42]. Esta herramienta fue desarrollada originalmente por investigadores de HP Labs utilizando lenguaje Java. Jena dispone de un conjunto de librerías para la manipulación de metadatos utilizados en la Web Semántica. Jena incluye además un motor de inferencia basado en reglas para realizar razonamiento basado en ontologías OWL y RDF y una variedad de

[42] https://jena.apache.org/

funciones para el almacenamiento de tripletas RDF en memoria o en disco. A continuación se realiza una descripción de los componentes desarrollados:

Aplicación para conversión de Objetos de Base de datos a Clase Ontología Local y Viceversa. Dentro de la aplicación es necesario el acceso a la base de datos en este caso se emplea un servicio Web donde se han definido los métodos *getDatosWSDL, changeObjectToOntology, changeOntologyToObject y loadOntGlobalToOntLocal.*

- El método *getDatosWSDL* permite la conexión mediante servicio Web al método *changeList* el cual retorna un listado de objetos de tipo Persona que en su atributo acción sea diferente de *OK* (significa que se han cambiado).
- El método *changeObjectToOntology*, tiene como parámetro de entrada un objeto (Persona) el cual será analizado y cada propiedad que contenga este será creada o actualizada según sea el caso.

- El método *changeOntologyToObject*, tiene como parámetro de entrada un objeto de tipo Modelo del cual se extraerá la información para la creación o actualización de los datos según sea el caso, la actualización se realiza mediante servicios Web.

- El método *loadOntGlobalToOntLocal*, tiene como parámetro de entrada un objeto tipo *OntModel*, en este método se realiza el proceso de conversión de la clase Ontología Global a la clase Ontología Local.

Aplicación para conversión de Clase Ontología Local a Clase Ontología Global. Para la comunicación entre aplicación locales y globales se ha implementado un servicio Web, el cual permite obtener y enviar los modelos ontológicos, a través del método *loadOntGlobalToOntLocal*.

7. ARQUITECTURA RECOMENDADA

Típicamente, las implantaciones de SOA son concebidas como proyectos a largo plazo, comúnmente, el tiempo de materialización del beneficio y la obtención de retorno sobre la inversión se dilatan en término de años. Sin embargo, con los procedimientos expuestos en la Sección 4, queda demostrada la factibilidad teórica y tecnológica de una arquitectura de integración rápida y con un esfuerzo no excesivo, siendo estas características clave para ganar la aceptación de los actores involucrados. La definición de esta arquitectura de integración de fuentes de datos se desarrolló siguiendo métodos actuales así como tecnologías de punta, tanto en el ámbito de las tecnologías semánticas así como con las tecnologías ESB.

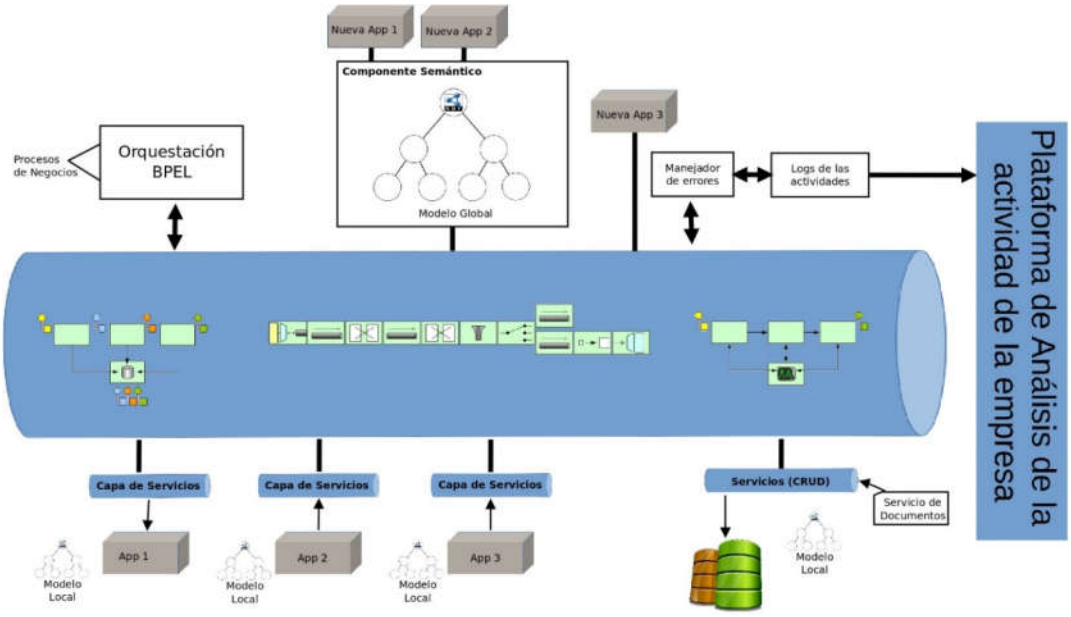


Figura 29. Arquitectura Recomendada

Como resultado final, en la Figura 29 se muestra la propuesta de integración, en donde el ESB posibilita la integración de fuentes de datos heterogéneas, la integración de cada fuente de datos implica el desarrollo de interfaces basadas en servicios Web que deben registrarse y enlazarse a las rutas definidas en el ESB, además se debe determinar el modelo ontológico local que represente las entidades y relaciones de la fuente de datos. Otro elemento que genera valor a

la propuesta es el mantenimiento del modelo ontológico global el cual sirve como punto de partida para la construcción de nuevas aplicaciones.

La importancia de esta arquitectura radica en que permite tener una visión futurista en cuanto al desarrollo o adquisición de nuevas aplicaciones. Teniendo en cuenta la arquitectura que se recomienda, ESB + semántica, esta quedaría preparada para evolucionar hacia una verdadera integración a nivel global utilizando los principios que marcan las tecnologías de la Web Semántica. Además, permitirá mantener un almacén de datos semánticos centralizado en donde se pueda realizar diferentes tipos de análisis de información o extracción de conocimiento utilizando técnicas de minería de datos.

Tomando en consideración esta arquitectura se puede planear la creación de nuevas aplicaciones basadas en la ontología global o directamente a través de las rutas definidas en el ESB. Por otra parte, se puede pensar en aplicación BPM[43] o BPEL[44] que pueden actuar sobre el ESB, afectando a todo el modelo de integración descrito en la propuesta.

La propuesta realizada, al ser concebida como “arquitectura” sin lugar a duda puede ser llevada a plataformas distintas a Apache Servicemix, ya sean estás de tipo propietario o de uso libre. Por otra parte, el resultado de este trabajo puede ser adoptado por cualquier organización que requiera integrar sus fuentes de datos, lógicamente es indispensable que las fuentes de datos a integrar sean compatibles con los estándares y protocolos que maneja el producto ESB seleccionado.

También se debe considerar que al incorporar el ESB se introducen latencias adicionales en la comunicación entre las fuentes de datos, razón por la que es necesario dimensionar adecuadamente los recursos computacionales, y en entornos donde se requiera alta disponibilidad se puede realizar el despliegue del ESB en una topología de clúster. Otro aspecto a considerar es la seguridad,

[43] https://www.ibm.com/developerworks/ssa/local/websphere/introduccion-bpm/index.html
[44] http://searchmicroservices.techtarget.com/definition/BPEL-Business-Process-Execution-Language

generalmente los ESB incorporan en su arquitectura mecanismos para gestión de seguridad, sin embargo, si se requieren características adicionales se pueden conectar componentes de seguridad que actúan como pasarela entre el ESB y las fuentes de datos.

Finalmente, los proyectos de integración de datos deben ser planteados y administrados con metodologías para gestión de proyectos de software en donde se aborden aspectos como: gestión de requisitos, costos, planificación, control de calidad, gestión de configuración, control de cambios, etc.

8. CONCLUSIONES Y RECOMENDACIONES

En este trabajo, se ha mostrado como la semántica juega cada día un papel más importante a la hora de satisfacer una determinada necesidad de información. Particularizando el problema de la semántica a la integración de datos provenientes de distintas fuentes heterogéneas entre sí, se ha visto como esta propuesta implica necesariamente el uso de algunas ontologías, tanto para modelar las aplicaciones locales, así como una ontología que permita modelar a la organización. Además, es necesario establecer las relaciones semánticas entre los modelos locales y el global. El modelo ontológico resultante debe representar todos los datos de las fuentes que se integren de modo que se pueda explotar todo su contenido para la generación de conocimiento. Estas consideraciones han sido recogidas en la propuesta de integración que permita un acceso uniforme de los datos provenientes de diferentes fuentes.

Por otra parte, se ha podido evidenciar que la infraestructura provista por el ESB favorece la interoperabilidad y es sin lugar a duda la tecnología más adecuada para llevar procesos de integración de fuentes de datos.

La combinación de ESB más semántica se presenta por tanto como una solución viable para obtener la integración efectiva de fuentes de datos en las organizaciones. Es necesario manifestar que la implementación desarrollada es un prototipo, que busca mostrar la viabilidad de este enfoque.

El trabajo futuro se orientará hacia el descubrimiento automático de los modelos que representen a una fuente de datos. Estos modelos permitirán a largo plazo un proceso de integración automático. Por otra parte, para demostrar la viabilidad de la propuesta, se planea realizar una experimentación sobre fuentes de datos en un entorno real. Finalmente, se pretende utilizar técnicas de procesamiento de lenguaje natural para establecer las relaciones entre los modelos ontológicos.

REFERENCIAS

Bernstein, P. A., & Haas, L. M. (2008). Information integration in the enterprise. Communications of the ACM, 51(9), 7279.

Beyer Mark A., Thoo Eric, Zaidi Ehtisham, Greenwald Rick, GARTNER. Magic Quadrant for Data Integration Tools, 2016. Recuperado de https://www.gartner.com/home.

Dittrich, K. R., & Jonscher, D. (2000). All Together Now: Towards Integrating the World's Information Systems. In JISBD (p. 7).

Echeverría, Daily & Astudillo, Hernán & Estrada, Rodrigo. (2008). ESB-QM: Modelo de Calidad para productos ESBs.

Harcuba O. and Vrba P., "Unified REST API for supporting the semantic integration in the ESB-based architecture," 2015 IEEE International Conference on Industrial Technology (ICIT), Seville, 2015, pp. 3000-3005.

Hohpe, G., & Woolf, B. (2003). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley.

Jin G. Y., Lv F. Z., and Xiang Z. Q., Using semantic technology to enhance ESB capabilities, Advanced Materials Research, vol. 849, pp. 298-301, 2014. Lecture Notes in Computer Science, vol 9622. Springer, Berlin, Heidelberg

Lenzerini, M. (2002, June). Data integration: A theoretical perspective. In Proceedings of the twentyfirst ACM SIGMODSIGACTSIGART symposium on Principles of database systems (pp. 233246). ACM.

Linthicum, D. S. (2000). Enterprise application integration. AddisonWesley Professional.

Roa-Valverde and J. Aldana-Montes, Extending ESB for semantic web services understanding, in On the Move to Meaningful Internet Systems: OTM 2008 Workshops (R. Meersman, Z. Tari, and P. Herrero, eds.), vol. 5333 of Lecture Notes in Computer Science, pp. 957 964, Springer Berlin Heidelberg, 2008.

Schratzenstaller W.M.K., Baldo F., Rabelo R.J. (2016) Semantic Integration via Enterprise Service Bus in Virtual Organization Breeding Environments. In: Nguyen N.T., Trawiński B., Fujita H., Hong TP. (eds) Intelligent Information and Database Systems. ACIIDS.

Shi K., Gao F., Xu Q. and Xu G., "Integration framework with semantic aspect of heterogeneous system based on ontology and ESB," The 26th Chinese Control and Decision Conference (2014 CCDC), Changsha, 2014, pp. 4143-4148

Suárez Figueroa, M. C., Gómez-Pérez, A., & Fernández-López, M.(2012). The NeOn methodology for ontology engineering. In Ontology engineering in a networked world (pp. 934). Springer Berlin Heidelberg.

Tsierkezos, S. A. (2010). Comparing Data Integration Algorithms (Doctoral dissertation, Thesis abstract. University of Manchester 2014.